République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université A.MIRA-BEJAIA

Faculté des Sciences Exactes
Département d'Informatique

THÈSE

Présentée par

Nadhir BOUKHECHEM

Pour l'obtention du grade de

DOCTEUR EN SCIENCES
Filière : Informatique

Option : Réseaux et Systèmes distribués

Thème

Time Synchronization in Wireless Sensor and Actuator Networks

Soutenue le : 26/06/2022 Devant le Jury composé de :

Nom et Prénom	Grade		
Mr. Abdelkamal TARI	Professeur	Univ. de Bejaia	Président
Mr. Nadjib BADACHE	Professeur	USTHB	Rapporteur
Hachem SLIMANI	Professeur	Univ. de Bejaia	Examinateur
Mr. Belaid AHROR	Professeur	Univ. de Bejaia	Examinateur
Djamel TANDJAOUI	Professeur	USTHB	Examinateur
Mohamed GUERROUMI	Maître de conférences (A)	USTHB	Examinateur

Année Universitaire : 2021/2022

People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University of Abderahmane Mira Béjaïa

Faculty of Exact Sciences
Department of computer science

Thesis

Presented by

Nadhir BOUKHECHEM

To obtain the degree of

Doctor of Science

Speciality: Computer Science

Option: Networks and Distributed Systems

On

Time Synchronization in Wireless Sensor and Actuator Networks

Thesis defended on 6/26/2022. The jury is composed of :

Chair:	Abdelkamal TARI	Professor, University A. Mira, Bejaïa.
Supervisor:	Nadjib BADACHE	Professor, USTHB.
Examiners:	Hachem SLIMANI	Professor, University A. Mira, Bejaïa.
	Belaid AHROR	Professor, University A. Mira, Bejaïa.
	Djamel TANDJAOUI	Research Director, CERIST.
	Mohamed GUERROUMI	Senior lecturer, USTHB.

Academic year 2021/2022.

To my parents, my wife, my children,
all my family and my friends.

Acknowledgment

First of all, I am grateful to ALLAH, the Almighty God, for giving me the strength and wisdom to complete this project.

I would like to express my gratitude to Professor Nadjib BADACHE, my thesis director, for whom I have a great deal of respect, for the precious help and the relevant advice he gave me throughout my project.

I would also like to deeply thank Professor Abdelmadjid Bouabdallah for welcoming me to the University of Technology of Compiègne. His advice helped me to obtain fruitful research results.

I express my best thanks to Pr. Abdelkamal TARI for his interest in my research work and for his acceptation to be the chairman in my thesis defense. I also want to thank Pr. Hachem SLIMANI, Pr. Belaid AHROR, Pr. Djamel TANDJAOUI and Pr. Mohamed GUERROUMI for their willingness to be reviewers of this thesis and members in the jury board. Personally, it is a great honor to have such prominent jury members.

Great thanks and gratefulness are due to my parents and family, especially my mother, grandmother, and aunt Fouzia, for their endless support, complete presence, and encouragement. I want also to express my appreciation to my wife for her patience, help, and encouragement especially in these moments.

Finally, I would like to thank all my friends and everybody who has helped me to achieve this humble work.

Nadhir Boukhechem

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

This chapter is a general introduction to this thesis. It gives an overview of the work presented in this manuscript and summarizes the different contributions.

1.1 Wireless sensor networks

A Wireless Sensor Network (WSN) is a set of inexpensive and autonomous sensor nodes equipped with wireless transmission interfaces. Usually, these sensor nodes are deployed in hundreds or thousands to cover a given territory and take measurements on their environment. These measurements are then transmitted to a base station for processing (see figure 1.1). To route the collected information to the base station, and since this station is often far from the sensors' transmission range, the sensors self-organize to form a multi-hop wireless network. Today WSNs are used in a variety of fields such as environmental monitoring [1], precision agriculture [2, 3], smart cities [4], and military applications [5, 6]. However, WSNs have several constraints. Indeed, the decentralized nature of these networks requires the use of distributed algorithms which are more complicated to develop than centralized ones. Also, the resources such as computing power, memory, and energy, must be used optimally. Currently, power consumption is the most important factor to consider in any WSNs application to maximize network lifetime and avoid crashes.

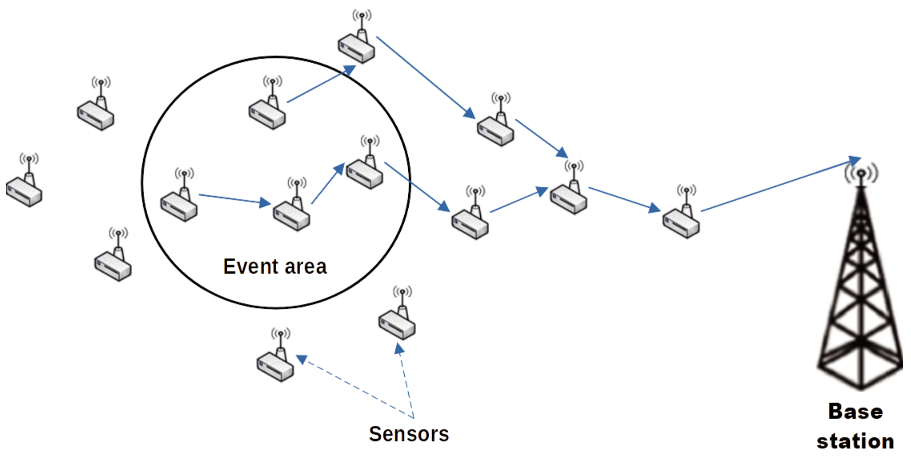


Figure 1.1: Wireless sensor network.

1.2 Wireless sensor and actuator networks

A wireless sensor and actuator network (WSAN) is a WSN with several actuators that can act on their environment. Unlike sensors, the actuators are rich in resources, indeed, they have more energy capacity and transmission range [82]. However, it should be noted that, due to their relatively high cost, there are typically much fewer actuators than sensors in the WSANs.

In WSANs applications, the sensors detect events in their environment and inform the actuators (see figure 1.2). These latter cooperate and take appropriate actions depending on the nature of the events. The WSANs are useful for monitoring environments exposed to critical conditions. For example, in forest monitoring applications [7, 8], when a fire starts, the sensors can detect the location of this fire and immediately inform the nearest actuators. These will then extinguish the fire before it spreads. Also, in the water distribution systems monitoring [9], sensors can detect water contamination and alert the actuators. Then, the actuators will react quickly to isolate the contaminated area and prevent its spread. Another application of WSANs is border surveillance and

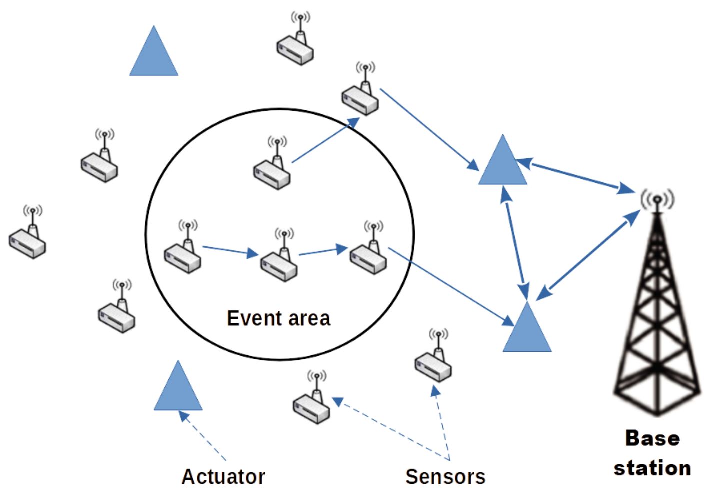


Figure 1.2: Wireless sensor and actuator network.

intruder detection. Indeed, the actuators can coordinate their actions to neutralize potential intruders detected by the sensors.

In any WSANs application, in addition to the WSN-related constraints, such as the limited resources of the sensors, and the multi-hop communication, some additional characteristics have to be considered. These characteristics include the following :

- *Nodes heterogeneity :* In WSANs, actuators have much more energy than sensors. This is an advantage since WSANs applications can put more load on the actuators and avoid using the sensors, which can increase the lifetime of the network. Another point is the transmission range of the actuators. Indeed, actuators are equipped with more expensive transmission antennas, which have a greater transmission range than sensor antennas. Obviously, this creates asymmetrical links between sensors and actuators. Unfortunately, most of the WSNs protocols do not support this characteristic and are no longer usable in WSANs. However, new protocols can take advantage of the actuators' transmission range to improve network performance. This is the case of our time synchronization protocol, called SanSync [9], which will be presented in chapter 4.

- *Communication delay constraints (Network latency):* An important characteristic of WSANs applications is that they require a fast response to events. For example, the actuators must coordinate their actions and respond immediately to extinguish a fire or prevent contamination. For this reason, the WSANs must guarantee a very short communication delay between nodes.

1.3 Time synchronization in WSANs

As seen previously, the actuators in WSANs applications need to cooperate and respond quickly to events. To coordinate their operations, the actuators require a common time reference used by all the nodes in the network (sensors and actuators). Indeed, the time reference is necessary to know exactly when an event has been detected by the sensors, and also to plan the actuators' actions. In this context, the purpose of time synchronization in WSANs is to provide an accurate common time reference to the network nodes while taking into account their specificities, particularly the limited resources of the sensors.

1.4 Thesis motivation

Accurate time synchronization of clocks is crucial for various WSNA protocols and applications. However, synchronizing the clocks is problematic due to the delays in delivering the synchronization messages, which are often non-deterministic. Currently, to our knowledge, most of the time synchronization protocols used to synchronize WSANs clocks are those developed for WSNs. Unfortunately, these protocols consider the actuators as simple sensors and do not take into account their additional resources. These additional resources can be counterproductive. For example, the difference in power between sensor and actuator antennas can generate asymmetric links between nodes, which hinders the proper functioning of WSNs protocols. However, it may be interesting to take advantage of the additional resources of the actuators to improve time synchronization accuracy. Our goal is to develop new time synchronization protocols specifically for WSANs. These protocols must exploit the available resources of the actuators, especially their extended transmission range, to increase time synchronization accuracy. This can indirectly improve the performance of WSANs applications that use the clocks.

1.5 Contributions

The scientific contributions of this thesis can be summarized as follows :

1. We developed a new time synchronization protocol for WSANs, namely Sensor and Actuator Networks Synchronization Protocol (SANSync), which organizes the network into clusters to fully exploit the transmission capacity of the actuators, and minimize synchronization errors. Indeed, inside the clusters, synchronization errors caused by the non-deterministic delays including the send time, the receive time, and the access time are eliminated, which is very advantageous compared to exiting protocols.

2. We developed another time synchronization protocol, namely Optimized Sensor and Actuator Networks Synchronization Protocol (OSANSync), which combines the strong points of SANSync and FCSA protocols. Indeed, OSANSync applies both an intra-cluster synchronization mechanism used by SANSync to synchronize the nodes inside the clusters, and a clock speed agreement mechanism used by FCSA to synchronize the nodes outside the clusters. This combination considerably improved time synchronization accuracy compared to SANSync and FCSA.

3. We proposed a heuristic method to select the best ROOT nodes for time synchronization in WSNs and WSANs. The ROOT node is the one that synchronizes all other nodes in the network. The proposed method is fully distributed and can be easily integrated with existing time synchronization protocols to improve their performance.

1.6 Publications arising from this thesis

The contributions of this thesis have resulted in the following publications :

1. Nadhir Boukhechem , Nadjib Badache. *SANSync: An Accurate Time Synchronization Protocol for Wireless Sensor and Actuator Networks.* Published in Wireless Personal Communications Journal, volume 105, pages951?972. (2019).

2. Nadhir Boukhechem , Nadjib Badache. *An optimized time synchronization protocol for WSANs.* Accepted in the 5th International Conference on Networking and Advanced Systems (ICNAS 2021).

3. Nadhir Boukhechem , Nadjib Badache. *A Fully Distributed Heuristic method to select the ROOT node in WSNs.* Submitted to the International Conference on Information Processing in Sensor Networks (IPSN 2022).

1.7 Thesis outline

The remainder of this thesis is organized as follows. chapter 2 introduce the theoretical concepts used in the manuscript. In chapter 3, we study the main time synchronization protocols developed for WSNs and WSANs. chapters 4 and 5 describe, respectively, the SANSync and OSANSync protocols and present the simulation results. In chapter 6, we propose a ROOT selection method, which has been tested in both WSNs and WSANs according to different parameters. Finally, we close up this thesis with a conclusion that summarizes the main contributions of our work and highlights some research perspectives and potential extensions for future work.

Chapter 2

Time synchronization in WSNs and WSANs

Contents

2.1 Introduction

In this chapter, we first present the theoretical concepts used overall in the thesis. Then, we explain the importance and difficulties of time synchronization in WSNs and

WSANs. Finally, we detail three fundamental mechanisms used to synchronize the clocks in these networks.

2.2 Time and clock

2.2.1 Coordinated universal time

The *coordinated universal time (UTC)*, also known as *real time*, is a very accurate time reference by which the world synchronizes clocks. In fact, the UTC is based on high-precision atomic clocks, which are the most precise and expensive clocks in the world. In the rest of this thesis, the UTC is called universal time, and we define a perfect clock as a clock that are synchronized with the UTC.

2.2.2 Physical clock

The *physical clock*, also known as *hardware clock* consists of an electronic oscillator (usually based on quartz crystal) and a register. The oscillator periodically generates electrical pulses which are counted in the register. In fact, each oscillator pulse corresponds to a predetermined time interval, and the time (hours, minutes, and seconds), shown by the clock, is calculated based on the register value.

The accuracy of a physical clock depends mainly on the stability of its oscillator. This stability is measured by calculating the ratio between its real and nominal frequency. For example, if the nominal frequency of the oscillator is 1 MHz while its real frequency is 1.0001 MHz, then it has a drift of 100 ppm (parts per million). The frequency of the oscillator depends on the form and the size of its quartz crystal which is very sensitive to climatic variations.

2.2.3 Logical clock

Generally, the physical clock rate and value cannot be modified by software. Therefore, the physical clock cannot be synchronized with the reference clock. A solution to this problem is to use a *logical clock* to determine the current time. This latter is calculated as a function of the physical clock value. Unlike the physical clock, the logical clock can be synchronized with a reference clock by changing its parameters. We take note that synchronized means that both clocks indicate the same time at any moment.

There are two options for synchronizing the logical clocks in a network :

1. *External synchronization* : it consists of synchronizing all network clocks with a reference clock outside the network, usually with the coordinated universal time. This approach is widely used in the literature.
2. *Internal synchronization* : it consists of synchronizing all network clocks with a reference clock inside the network. The clocks in this approach can also be synchronized with each other, by consensus, without using any reference clock.

2.2.4 Events ordering

In distributed systems, we also refer to a logical clock, which differs from the one seen above. This logical clock does not indicate the time but allows the global ordering of events. Indeed, it is often not necessary to know when an event has occurred, but only to determine the order of events. Among the most well-known algorithms for events-ordering are the Lamport clock [84] and Vector clock [85].

2.2.5 Physical clock and logical clock in WSNs and WSANs

The sensors and actuators in WSANs are usually equipped with inexpensive clocks. However, the accuracy of these clocks varies by manufacturer. Typically, the sensor and actuator clocks exhibit a drift between 30 and 100 ppm [19]. In WSANs, the physical clock of a node i at universal time t , denoted $H_i(t)$, is generally defined as follows:

$$H_i(t) = \int_{t_0}^{t} h_i(\tau)\, d\tau \tag{2.1}$$

Where t_0 is time when the physical clock of node i started. h_i represent the rate (speed) of the physical clock. The clock drift is assumed to be bounded [19 ,17]. So, for each time t, we have:

$$1 - \varepsilon \leq h_i(t) \leq 1 + \varepsilon \text{ , } (0 < \varepsilon \ll 1) \tag{2.2}$$

Each node i in the WSANs, besides its physical clock $H_i()$, maintains a logical clock denoted $L_i()$. This latter is calculated each time a node i read its logical clock. The logical clock of a node i at universal time t is defined as follows:

$$L_i(t) = \int_{t_0}^{t} l_i(\tau)\, h_i(\tau)\, d\tau + \theta_i(t_0) \tag{2.3}$$

Where $\theta_i(t_0)$ represents the logical clock offset of node i, at universal time t_0, with respect to the reference clock. l_i is called *rate multiplier*. Modifying the value of l_i allow to accelerate or reduce the speed of the logical clock.

The offset of a clock $L_i()$ with respect to another clock $L_j()$ at universal time t is defined as follows: $L_i(t) - L_j(t)$. Also, the offset of a clock $L_i()$ with respect to a perfect clock, at universal time t, is defined as follows: $L_i(t) - t$. This last offset is illustrated in figure 2.1. It should be noted that in the remainder of this thesis, the term "clock" refers to the logic clock.

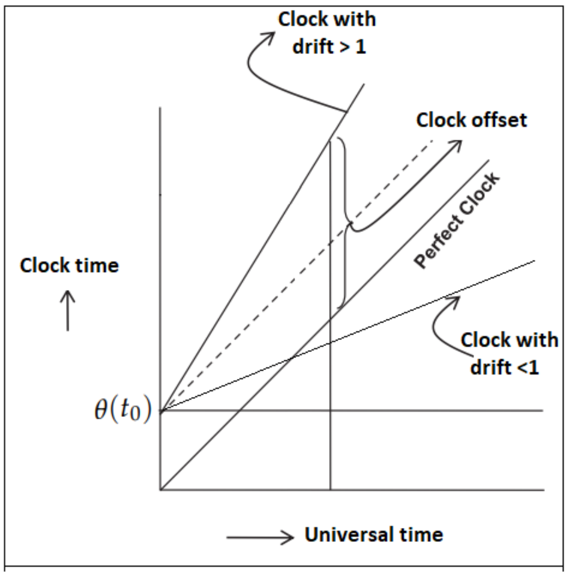


Figure 2.1: Clock offset illustration

2.3 Importance of time synchronization in computer networks

Time synchronization plays an important role in networks and distributed systems. Indeed, many critical services use different timestamps to ensure their operation. Here are some examples :

- *Log files and monitoring :* A log file records chronologically the events that have occurred. For example, they record the date and time of connections, system errors, machine and router failures, and user actions. This information can be used to understand the source of errors and correct them. It is also used by administrators to provide statistics and monitor networks. However, to exploit the information contained in the log files, accurate timestamps are required.

- *Accurate Transaction :* Time synchronization is crucial in networks to perform simultaneous commercial transactions. Indeed, when systems are geographically distant, an accurate time reference is needed to execute the request in a correct sequence.
- *Schedule operations :* Sometimes the system administration requires the scheduling of certain tasks that need to be performed at specific times, for example, data backup and recovery. In this case, time synchronization between machines becomes crucial to ensure that the scheduled tasks are properly coordinated.
- *Access Security and Authentication :* Currently, the authentication protocols use domain time in their authentication processes. The timestamps used in a session between two machines are considered valid only if the time difference between them is less than the maximum time difference specified in the protocol policy.

2.4 Importance of time synchronization in WSNs and WSANs

Various WSNs and WSANs applications and protocols require the presence of an accurate time reference to function correctly. Below are some examples :

- *Node localization:* Many localization protocols use the transmission delay of a radio signal, and its propagation speed, to calculate the distance between nodes in the WSANs, then they use it to determine the position of each node. However, the accuracy of the time reference used by these localization protocols has a direct influence on their performance [14], it also affects related applications such as detection and tracking systems.
- *Data aggregation :* In WSANs, data aggregation consists of grouping sensors data into the same packets. The purpose of this is to minimize the number of packets transmitted by the sensors and thus minimize energy consumption. For example, we can merge humidity and temperature data that were collected in the same place at the same time, to avoids data redundancy. In this case, an accurate common time reference is necessary to determine the temporally-related data.
- *Sleep scheduling :* A scheme for energy conservation in WSANs is to periodically disable network nodes. To do this, all nodes in the WSANs must switch on and

off at specific times based on a common time reference. This scheme is suitable for applications that do not require continuous environmental monitoring such as precision farming.

- *Transmission channel access:* A method of accessing the transmission channel in wireless networks is TDMA (Time-division multiple access). This method allows multiple traffic streams on a single frequency band. Indeed, TDMA distributes the available time between the different nodes, each one receives a fraction of the time. However, TDMA requires an accurate time reference to avoid collisions and ensure equitable exploitation of the transmission channel.

2.5 Difficulties of time synchronization in WSNs and WSANs

The physical clocks of the sensors and actuators use crystal oscillators to generate the clock's signals. The frequency of these oscillators determines the clock's speed. We note that this frequency is not stable due to manufacturing imperfections, and environmental changes, such as temperature, pressure, battery voltage, etc [11, 12, 13]. Therefore, the physical clocks of the sensors and actuators do not run at the same speed, and need to be periodically re-synchronized.

To explain the difficulties of time synchronization in WSNs and WSANs, we consider two nodes A and B and assume that node A has to synchronize the clock of node B. To do this node A send a message containing its clock value to node B, this one will then synchronize its clock according to the received value. However, the main problem that arises here is that before node B receives the clock value of node A, this one changes because of the delay between sending the message by node A and receiving it by node B (figure 1.1). This delay has been studied in [15, 16, 17,18], and it is decomposed as follows :

- *Send time :* it is the time the operating system takes to prepare and send the message. The send time is non-deterministic.
- *Access time :* it is the time the node A takes to access the transmission channel, it depends on the availability of the channel which is shared between multiple nodes. The access time is non-deterministic.

- *Transmission/Reception time :* the transmission time is the time needed for transmitting the message by the antenna of node A. On the other side, the reception time is the time needed for receiving the message by the antenna of node B. Unlike the previous delays, the transmission/reception time is deterministic and depends on the message size and data rate.
- *Propagation time :* it is the time the message takes to travel from node A to node B. The propagation time is deterministic and depends on the distance between the two nodes.
- *Receive time :* it is the time needed for receiving the message by node B. This delay is similar to the send time.

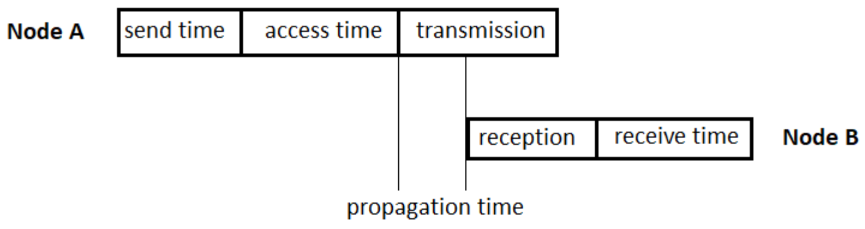


Figure 2.2: Transmission delays between two nodes

Another characteristic of WSANs, which complicates more the time synchronization, is the multi-hop communication between the nodes. Indeed, the synchronization message between node A and node B may eventually pass through intermediate nodes. This generates additional delays that must be taken into account.

2.6 Time synchronization mechanisms in WSNs and WSANs

The time synchronization protocols in WSNs and WSANs use three basic mechanisms to synchronize the clocks, including the sender-receiver, the receiver-receiver, and the sender synchronization mechanism [17, 20, 24]. These mechanisms are explained below.

2.6.1 Sender-receiver synchronization mechanism

The sender-receiver synchronization mechanism uses a bi-directional exchange of time information between nodes to synchronize their clocks. A well-known example of sender-receiver protocols is the Timing-sync Protocol for Sensor Networks (TPSN) [34].

To synchronize the nodes' clocks, TPSN constructs a spanning tree from a given *reference node*. At the start, the reference node synchronizes the clocks of its son nodes. Then, the synchronized nodes, in turn, synchronize the clocks of their son nodes in the tree. This process continues until all the node clocks in the network are synchronized. TPSN use a *pairwise synchronization* scheme (figure 2.2). Indeed, if a node A has to synchronize the clock of a son node, say B, it first sends to node B a start message to initiate the synchronization process. After that, node B sends to node A a synchronization message containing its clock value $T1$. Node A receives the message at time $T2$. Then, node A sends a response message to node B containing values $T1$, $T2$, and $T3$, where $T3$ is the clock value of node A when it sends the response message. At last, node B receives the response message at time $T4$ and calculates its clock offset as follows:

$$Clock\_offset = [(T2 - T1) - (T4 - T3)]/2$$

$$Where:$$

$$T2 = T1 + Propagation\_Time + Clock\_Offset$$

$$T4 = T3 + Propagation\_Time - Clock\_Offset$$

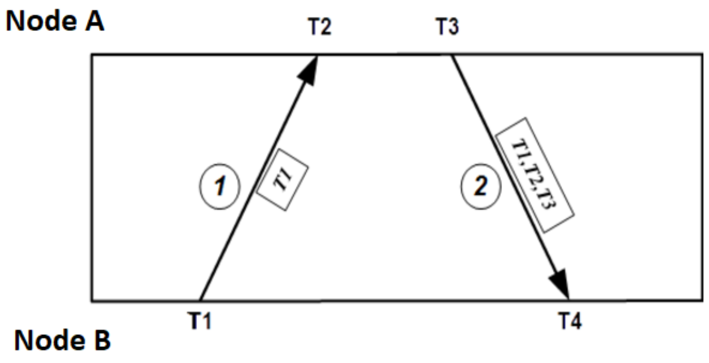


Figure 2.3: Pairwise synchronization

We note that the pairwise synchronization supposes that bidirectional message exchange between two nodes is temporally symmetric.

Since the velocity of the electromagnetic wave is very high, the propagation time of the messages between nodes A and B is negligible and does not have a great influence on time synchronization accuracy. Contrary to the send time, the receive time, and the access time, which must be considered.

2.6.2 Receiver-receiver synchronization mechanism

The receiver-receiver synchronization mechanism takes advantage of the broadcast nature of the wireless medium to synchronize the clocks of a set of receiver nodes simultaneously (see Figure 2.3). A well-known example of receiver-receiver protocols is the Reference Broadcast Time Synchronization (RBS) [36]. The latter allows synchronizing the clocks of several receiver nodes that are in the transmission range of a sender node. We note that the sender node clock is not synchronized. To do this, the sender node in RBS first broadcasts a beacon to the receiver's nodes. The reception time of the beacon is considered as a reference by the receiver nodes. Then, each receiver records the reception time of the beacon and sends it to the other receivers. At last, each receiver estimates its clock offset with respect to other receiver node clock values. RBS assumes that the antennas of the receiver nodes receive the beacon at the same time. In this way, the time synchronization errors due to the send time and the access time delays are eliminated. Indeed, the only remaining source of error is the receive time. We note that the authors of RBS also proposed an extension of this protocol to a multi-cluster-based network. Another protocol called R4Sync[83] includes timestamps with the beacons and distributes the referencing function among all nodes. This eliminates the single point of failure (the sender node), the weak point of RBS, and reduces the overhead.

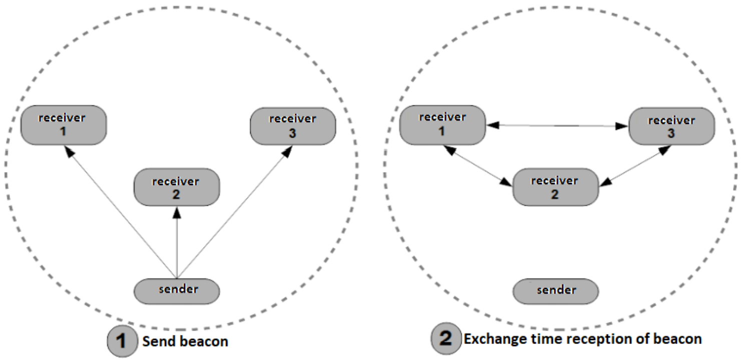


Figure 2.4: Reference Broadcast Time Synchronization

2.6.3 Sender synchronization mechanism

The sender synchronization mechanism use unidirectional synchronization messages to synchronize the node clocks. A well-known example of sender protocols is the Flooding Time Synchronization Protocol (FTSP) [15], which allows synchronizing all the node clocks in the network with the clock of a *reference node*. The latter periodically broadcasts a synchronization message containing its clock value to nodes in its transmission range. Then, each receiver node records the time reception of the message and the clock value of the reference node as a pair in a least-squares table. Each entry in the least-squares table provides a *synchronization point*. After recording enough points in its least-squares table, each receiver node uses the least-squares regression to adjust its clock offset and drift with respect to the reference node clock. After being synchronized, each node synchronizes, in turn, the nodes' clocks in its transmission range using the same mechanism, and so on until all the nodes' clocks in the network are synchronized.

In FTSP, the sender node records its clock value in the MAC layer just before transmitting the synchronization message. This eliminates the send time and the access time delays. Indeed, the only remaining source of error is the receive time. By estimating clocks speeds, FTSP avoids frequent re-synchronization of the clocks, which reduces the time synchronization overhead.

2.7 Conclusion

In this chapter, we have shown the importance of time synchronization in networks, in particular in WSNs and WSANs. However, synchronizing the clocks is constrained by the non-deterministic delays in the transmission of the synchronization messages. Also, due to the limited resources of the sensors, synchronization methods used in traditional networks, such as NTP (Network Time Protocol) and GPS (Global Positioning System), are not suitable for WSNs and WSANs. In the next chapter, we will examine the different time synchronization protocols developed for WSNs and WSANs.

Chapter 3

Time synchronization protocols in WSNs and WSANs

Contents

3.1 Introduction

This chapter focuses on the literature review. First, we describe in detail the time synchronization protocols for WSNs. These have been classified according to different aspects. Next, we present the time synchronization protocols for WSANs. Currently, very few protocols have been proposed exclusively for WSANs. Finally, we conclude the chapter by indicating the type of protocols suitable for WSAN.

3.2 Time synchronisation protocols in WSNs

3.2.1 Protocols based on spanning trees

Many protocols in literature use spanning trees to synchronize the node clocks in WSNs [16, 25, 26, 70]. Greunen and Rabaey [16] proposed the lightweight tree-based synchronization protocol (LTS). As in TPSN, LTS builds a spanning tree from a given reference node and uses the pairwise synchronization scheme to periodically synchronize the node clocks. However, in LTS a new spanning tree is created for each new synchronization session. This allows adapting the spanning-tree if the network topology changes.

Due to the drift of the node clocks, they must be periodically re-synchronized. In LTS the re-synchronization interval can be adapted according to the accuracy required by applications. We note that the synchronization accuracy of a node clock depends mainly on the number of hops that separate it from the reference node and on the clock drift. In LTS, each time the nodes are synchronized, the reference node determines the maximum depth of the spanning tree used to synchronize the node clocks. Based on this parameter, and on the required accuracy, and on the maximum drift of the clocks, LTS adjusts the re-synchronization interval. Therefore the overhead in LTS is relative to the application needs, which is an advantage compared to previous protocols.

Rahamatkar and Agarwal [35] proposed a Reference Based, Tree-Structured Time Synchronization Approach (TSRT). Unlike TPSN and LTS, This protocol assigns for each node a unique clock channel different from all its neighbors, which reduces the variation in the transmission delay. Also, TSRT allows the use of several reference nodes.

Zhehan Ding and Yamauchi [25] designed another synchronization protocol for WSNs based on a spanning tree. The latter is built from a given reference node. The objective of this protocol is to reduce power consumption and increase the lifetime of sensor nodes rather than to increase the accuracy of time synchronization. To do this, the reference node selects its child node that has the most energy, e.g. node 1. Then, the clock of node 1 is synchronized using a pairwise synchronization scheme. The reference node uses broadcast messages to communicate with node 1. So the other child nodes of the reference node also receive the messages transmitted to node 1. Then, these messages are used by the child nodes to synchronize their clocks. We note that only the reference node and node 1 transmit messages. Indeed, the child nodes only receive the messages transmitted by the reference node. This preserves the energy of the nodes and increases their lifetime. To explain the synchronization scheme of this protocol, we consider a spanning tree composed of a reference node and three child nodes (nodes 1, 2, 3). We note that this scheme can be used in the case of several child nodes. Figure 3.1 illustrate the exchange of synchronization messages. We assume that node 1 has more energy than nodes 2 and 3. In our case, the clocks synchronization is performed as follows:

1) At time $t1$, the reference node broadcast to its child nodes (nodes 1, 2, 3) a synchronization message containing the identifier of node 1. It's assumed that the message will be received at the same time by the child nodes.

2) After Node 1 receives the synchronization message, it records the receiving time $t2_{Node1}$ and sends back a message to the reference node containing the values $t2_{Node1}$ and $t3_{Node1}$. This latter value is the time when node 1 sends back the message.

3) When the reference node receives the message from Node 1, it records the receiving time $t4$ and calculate the time offset $\theta_{ref,Node1}$ with Node 1 as follows :

$$\theta_{ref,Node1} = [(t2_{Node1} - t1) - (t4 - t3_{Node1})]/2. \tag{3.1}$$

After calculating the offset, the reference node broadcast a message to its child nodes containing the values $\theta_{ref,Node1}$ and $t2_{Nodel}$.

4) When nodes 2 and 3 receive the message from the reference node, they calculate their time offsets $\theta_{ref,Node2}$ and $\theta_{ref,Node3}$ as follows :

$$\theta_{ref,Node2} = \theta_{ref,Node1} - (t2_{Node1} - t2_{Node2}), \tag{3.2}$$
$$\theta_{ref,Node3} = \theta_{ref,Node1} - (t2_{Node1} - t2_{Node3}). \tag{3.3}$$

In this schema, nodes 2 and 3 synchronize their clocks without sending any message, which saves their energy.

Tie Qiu et al. [26] proposed the Spanning Tree-based Energy-efficient Time Synchronization (STETS) protocol. The latter uses both sender-receiver and receiver-receiver

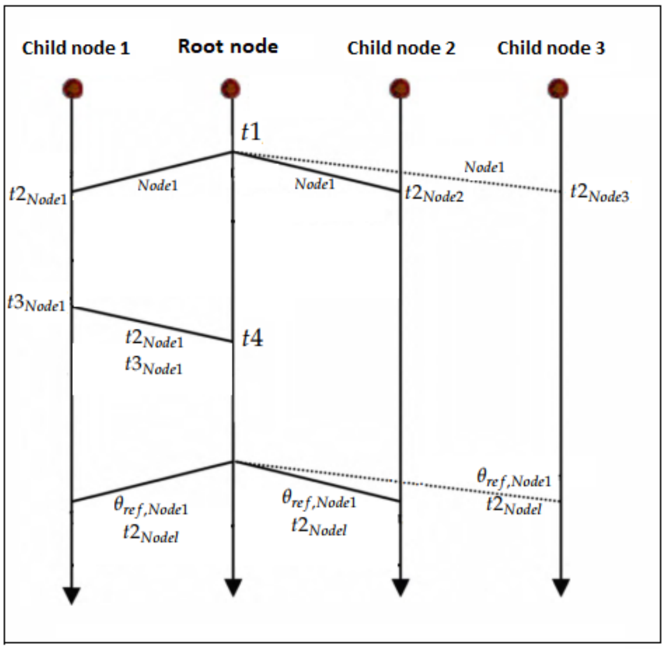


Figure 3.1: Broadcasting synchronization based on spanning tree

synchronization mechanisms. In STETS, backbone nodes form a spanning tree and synchronize their clock using a sender-receiver mechanism. The Other nodes are synchronized using a receiver-receiver mechanism. The strong point of STETS is in the way he builds the spanning tree. Indeed, this protocol ensures that only small number of nodes (backbone nodes) form the spanning tree. These nodes have the right to send synchronization messages. By contrast, the other nodes (passive nodes) do not belong to the spanning tree, and they only listen to backbone nodes to synchronize their clocks. Figure 3.2 gives an example of a spanning tree formed by STETS.

The approach used by STETS is advantageous for densely connected network as it reduces communication overheads while maintaining a high level of accuracy. However, there is a defect in the construction phase of the spanning tree. Indeed, some nodes (the isolated nodes) are out of the transmission range of the backbone nodes, and remains unsynchronized. To solve this problem, the same authors of STETS designed the R-Sync protocol [27]. The latter adds a phase to detect the isolated nodes. However, this phase, which is executed before starting the time synchronization, involves an additional overheads.

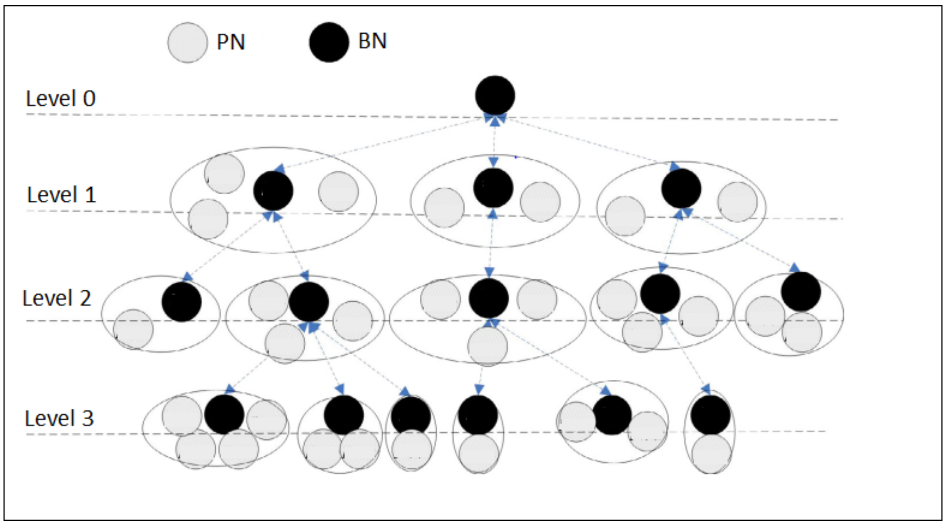


Figure 3.2: Spanning tree structure formed by STETS [26].

3.2.2 Protocols based on clustering

In cluster-based protocols, such as [21, 28, 29, 30 32, 33, 71, 72, 73, 74], the network is organized into clusters. Each one is composed of a cluster head and several cluster members. The cluster heads generally synchronize their clocks with a reference clock or with other cluster head clocks. And the cluster members synchronize their clocks with the associated cluster head clock. Thus, using clustering has the advantage of prolonging the network lifetime and offering better scalability [32, 42].

The Scalable Lightweight Time-synchronization Protocol for wireless sensor networks (SLTP) [28] organizes the network into cluster heads, cluster members, and gateways nodes. In this protocol, a random node in the network becomes a cluster head and broadcasts its status to its neighbors. Each node that receives the status message from the cluster head becomes a cluster member and broadcasts its new status to its neighbors. Next, each neighbor that receives a status message from a cluster member, and doesn't have a status yet, becomes a cluster head. This procedure is continued until the whole network is covered. The nodes that receive status messages from more than one cluster head become gateway nodes. Each gateway node will inform its cluster heads that it has become a gateway. It will also provide a list of its neighboring cluster heads. This allows cluster heads to select the gateways between clusters.

After the cluster construction phase, each cluster head broadcasts several synchronization messages containing its clock value to its cluster members. The latter, after receiving several clock values, used the linear regression method to adjust their clock offset and drift with respect to the cluster head clock. Note that the gateways synchronize with multiple cluster heads. SLTP assumes that cluster members communicate only with their corresponding cluster heads and do not need to communicate with each other. Therefore, SLTP does not employ a global time. Indeed the nodes use local times (cluster head times) to timestamp events. If a packet containing a timestamp is transmitted through different clusters, this timestamp is converted by the gateways to the local cluster times. The accuracy of SLTP method is the same as RBS, but for large area and long life clusters SLTP is a better choice [28].

A protocol similar to SLTP is the L-SYNC time synchronization protocol [29]. The latter used the same synchronization method as SLTP except that L-SYNC selects the cluster heads based on the degree of the nodes (the number of their neighbors). This minimizes the number of cluster heads and therefore minimizes cluster overlap. Indeed, fewer clusters and overlaps will decrease the channel competition between clusters and improve algorithm efficiency [29].

The Cluster-Based Time Synchronization Protocol for Wireless Sensor Networks (CTS) [30] used a multi-level clustering method [31] to organize the network in a top-down hierarchical structure. In CTS the cluster heads are organized into different levels and synchronized using the pairwise synchronization scheme as in the TPSN protocol. To do that, the sink (level-0) synchronizes all the clocks of the level-1 cluster heads. Then, the level-1 cluster heads in turn synchronize the clocks of the level-2 cluster heads, and so on until all the cluster head clocks are synchronized. To synchronize the cluster member clocks, each cluster head broadcasts its clock value to its cluster members. These latter adjust their clocks according to the received value. The CTS protocol, unlike SLTS and L-SYNC protocols, synchronizes all the node clocks of the network to the sink clock (global synchronization), so no time conversion is needed. However, the time synchronization accuracy remains dependent on the depth of the hierarchical topology used to synchronize the cluster heads.

The protocol (Synchronization through Piggybacked Reference Timestamps) [32] tries to minimize the power consumption of the time synchronization in clustered WSNs. As in CTS protocol, the cluster heads in SPiRT are synchronized using the pairwise synchronizations schema. However, SPiRT profit of the synchronization phase of the cluster heads to also synchronize the cluster members. Indeed, the cluster members only catch the synchronization messages sent by their cluster head and used the timestamps containing in these messages to synchronize their clocks. We note that is not necessary for the cluster members to capture the synchronization messages sent to their cluster head. Also, it is not necessary to send additional synchronization messages to the cluster members. This cuts on energy consumption and increases the synchronization efficiency of SPiRT [32]. We note that SPiRT adjusts clock offset and drift using the Maximum Likelihood Estimators (MLE) and the corresponding CramerRao Lower Bounds (CRLB).

3.2.3 Protocols based on flooding

The protocols based on flooding, such as [19, 38, 50, 79], do not require any organization of the network topology. To synchronize the node clocks, these protocols disseminate time information through the network via flooding. This makes them robust to node mobility and failure. A well-known protocol in this category is FTSP which is described previously. However, the latter exhibits a synchronization error that grows exponentially with the network diameter [19, 38]. This is due to errors in estimating clock drifts, and the fact that the nodes in FTSP do not immediately broadcast the clock value of the reference node. Indeed, in FTSP, each node uses a periodic timer to determine when the data is broadcasted. This amplifies the clock estimation error at each hop. To overcome this problem, PulseSync protocol [39] floods the network with short, fast pulses. The time information from the reference node is propagated as fast as possible. Also, PulseSync uses an experimental test to estimate the mean of pulse delay and compensate for the clock offset estimation. Consequently, the synchronization error grows with the square root of the network diameter rather than being exponential. However, one of the drawbacks of PulseSync is that the rapid flooding in WSN can also be slow due to neighborhood contention because the nodes cannot propagate the flood until their neighbors have finished their transmissions [19]. Also, reliable rapid flooding in sensor networks is difficult due to packet losses [19].

The FCSA [19] protocol employed a clock speed agreement method that forces all node cocks in the network to run at the same speed. This reduces the amplification of synchronization errors at each hop while allowing periodic broadcasting of time information. The synchronization error in FCSA grows with the square root of the network diameter. The clock speed agreement method is based on averaging the time information received from neighbors, it will be detailed in chapter 5. FCSA exhibits similar performance to PulseSync on the line topology, but it is superior on the grid topology where contention and congestion are not negligible [19].

Shi et al. [50] proposed a rapid flooding multiple one-way broadcast time-synchronization (RMTS) protocol for large-scale wireless sensor networks. In the latter, several synchronization packets are broadcasted by the nodes at a very short time in a single synchronization period. After several synchronization periods, the clock drift and the clock offset of each node are calculated using the maximum likelihood estimation (MLE). By broadcasting multiple packets in a single period, RMTS ensures rapid convergence and improved accuracy. However, this generates an additional load on the network, so RMTS consumes more energy.

3.2.4 Protocols based on consensus

The consensus-based protocols are fully distributed and do not require a reference node or gateways. Indeed, the nodes' clocks are synchronized with each other by exchanging synchronization messages between neighboring nodes. Figure 3.3 illustrates how synchronization messages are exchanged in a network composed of 7 nodes. Figure 3.3(a) represents the protocols that synchronize the clocks from a reference node (Node 0), such as FTSP and TPSN. On the other hand, figure 3.3(b) represents the consensus-based protocols where no reference node is required. In this case, the synchronization messages are exchanged between neighboring nodes.

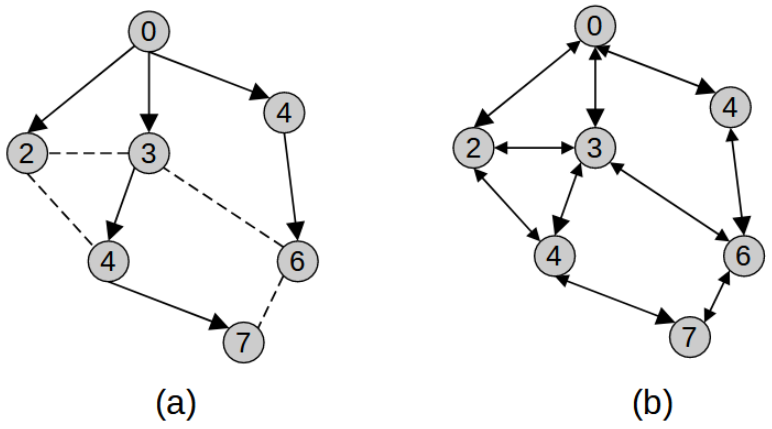


Figure 3.3: Reference node (a), and consensus based (b) synchronisation protocols

Many protocols in literature are based on consensus algorithms [40, 42, 43, 44, 45, 46, 47, 75, 76, 77, 78]. In gradient time synchronization protocol (GTSP) [40], each node periodically broadcasts to its neighboring nodes a synchronization message containing its current logical clock value $L_i(t)$ and its rate multiplier $l_i(t)$. Each node i that receives clock values from its neighbors, updates its own clock as follows :

$$\theta_i(t_{k+1}) = \theta_i(t_k) + \frac{\sum_{j \in N_i} L_j(t_k) - L_i(t_k)}{|N_i| + 1}. \tag{3.4}$$

After several iterations, all the logical clocks converge to the same value.

GTSP also define the absolute logical clock rate, noted $xi(t)$, of a node i at universal time t as follows:

$$x_i(t) = h_i(t)\ l_i(t). \tag{3.5}$$

The logical clock is therefore defined as follows :

$$L_i(t) = \int_0^t x_i(\tau)\, d\tau + \theta_i(t_0). \tag{3.6}$$

To synchronize the nodes' clocks, the absolute logical clock rate of each clock $L_i()$ is iteratively updated as follows:

$$x_i(t_{k+1}) = \frac{\left(\sum_{j \in N_i} x_j(t_k)\right) + x_i(t_k)}{|N_i| + 1}, \tag{3.7}$$

where N_i is the set of neighbors of node i.

Theoretically, all the absolute logical clock rates will converge to the same value. However, in practice, nodes cannot estimate their hardware clock rate h_i and consequently cannot update their absolute logical clock rate $x_i(t)$. To overcome this problem, each node i in GTSP uses the following equation to update its clock rate multiplier. After several iterations, all the absolute logical clock rates will converge to the same value.

$$l_i(t_{k+1}) = \frac{\left(\sum_{j \in N_i} (x_j(t_k)/h_i(t_k))\right) + l_i(t_k)}{|N_i| + 1}. \tag{3.8}$$

GTSP has the advantage of being fully distributed and robust to node failures. Also, GTSP achieves a better local synchronization. Indeed, in the protocols seen previously such as FTSP et TPSN, the geographically close nodes can be far in terms of path-length in the constructed tree. This has a direct impact on synchronization errors between these nodes. This is particularly harmful to many applications such as target tracking or time division medium access (TDMA) scheduling, for which it is really important that clock errors between one node and the others degrade sufficiently smoothly as a function of geographic distance [42].

The External Gradient Time Synchronization Protocol (EGSync) [41] aims to minimize the synchronization error between nodes that are geographically close and at the same time allows to synchronize the nodes' clocks with a predefined reference node. Indeed, the consensus-based protocols seen above do not allow a global synchronization with a specific clock or an external clock. To do this, each node j in the network periodically broadcasts to its neighbor nodes a synchronization message containing its physical clock value H_j, its logical clock value L_j, its logical clock rate multiplier l_j, the latest rate multiplier of the reference node clock l_j^{ref}, and the latest difference between the reference node logical clock and the reference node physical clock Δ_j^{ref}. Each time a node i receives a new synchronization message from a neighbor node j, estimate the relative physical clock rate h_i^j of the neighbor node j using the least-squares regression method. We note that each node i in the network maintains a repository to keep track of the relative physical clock rates and rate multipliers of its neighbors. These values are then used by the node i to update its rate multiplier li as follows:

$$l_i(t^+) = \frac{l_i(t) + \sum_{j \in N_i} h_i^j(t)\, l_i^j(t)}{|N_i| + 1}, \tag{3.9}$$

where t^+ represents the time immediately after the update operation. N_i and $|N_i|$ represent the set and the number of neighbor nodes of node i, respectively.

Also, node i update its logical clock offset $\theta_i(t^+)$ as follows:

$$\theta_i(t^+) = \theta_i(t) + \frac{\sum_{j \in N_i}(L_i^j(t) - L_i(t))}{|N_i| + 1}, \tag{3.10}$$

where $L_i^j(t)$ represents an estimation of the logical clock value of node j.

With this execution, it can be proven theoretically that all nodes agree on a common logical clock speed and value. After that, for all times t, for each node i, we have :

$$h_{ref}(t) = h_i(t)\frac{l_i(t)}{l_{ref}(t)}. \tag{3.11}$$

The logical clock of node i is therefore calculated as follows :

$$L_i(t) = \int_0^t h_i(\tau)\frac{l_i(\tau)}{l_i^{ref}(\tau)}\,d\tau + \theta_i(t). \tag{3.12}$$

Each node i can also calculate the hardware clock value of the reference node H_v^{ref}, as follows:

$$H_v^{ref}(t) = L_i(t) + \Delta_i^{ref}(t). \tag{3.13}$$

In this way, EGSync allows at the same time to have a good local synchronization while allowing a global synchronization with a reference clock.

Schenatoa and Fiorentin [42] designed a consensus-based protocol for clock synchronization in wireless sensor networks (ATS: Average TimeSynch). Unlike GTSP and EGSync protocols, ATS is asynchronous, which makes it more resistant to packet loss and node failure. To achieve this, when a node j in ATS transmits a packet in the network this latter contains the node identifier id_j, its physical clock value H_j, its logical clock rate multiplier l_j, and its logical clock offset θ_j. Each time a node i receive a new message from node j, it stores the received time H_i^j (the physical clock value of node i), and updated its logical clock offset as follows:

$$\theta_i(t+) = \theta_i(t) + (1 - p_o)(H_j(t) - H_i(t)), \tag{3.14}$$

where p_o is a design parameter that can be set between 0 and 1.

Also node i estimate the rate multiplier l_i^j relative to the physical clock of node j, and updated its logical clock rate multiplier l_i as follows :

$$l_i^j(t+) = \frac{H_j - H_j^{(old)}}{H_i^j - H_i^{j(old)}}, \tag{3.15}$$

$$l_i(t+) = p_v\, l_i(t) + (1 - p_v)\, l_i^j(t+)\, l_i(t), \tag{3.16}$$

where p_v is a design parameter that can be set between 0 and 1. In practice, the necessary conditions for asymptotic convergence are that the communication graph is connected and that each node transmits sufficiently often [42].

Lamonaca et al. [45] proposed a clock synchronization protocol for wireless sensor networks with a selective convergence rate. The latter is based on a revised version of ATS protocol. The proposed protocol makes a compromise between synchronization accuracy and network life-time by increasing or decreasing the rate of exchange of the synchronization messages. Indeed the protocol focuses on the parts of the network where events are detected. And assumes that only the nodes in that part require a high synchronization accuracy. This is the case for measurement applications. This can significantly preserve the node energy and consequently increase the lifetime of the network. However, this approach is limited to specific applications such as environmental measurement systems.

Wu et al. [43] proposed the clustered consensus time synchronization protocol (CCTS). The latter combines the consensus-based time synchronization approach with clustering. This to obtain faster convergence and better energy efficiency. To do that, CCTS uses intra-cluster and inter-cluster time synchronization. In the intra-cluster synchronization, each cluster-head exchanges multiple messages with its cluster members to estimate the cluster clock's compensation parameters (to compensate the average clock offsets and the average clock drifts within the cluster). These compensation parameters are then transmitted to the cluster members to update their clocks. After that, in the inter-cluster time synchronization, the cluster-heads exchange with each other the compensation parameters and attribute to each one a weight according to the size of each cluster. In the end, the cluster-heads updates their clocks according to the received parameters. This process is repeated until all the clocks in the network converge to the same value and the same speed.

The network topology is an important parameter that affects the convergence of the consensus-based protocols. Indeed, the more the degree of connectivity is, the more the convergence is fast. Panigrahi and Khilar [44] designed a multi-hop consensus time synchronization protocol, called SATS (multi-hop Selective Average Time Synchronization). Unlike the previous consensus-based protocols that used single-hop information (direct neighbor information) to update their clock, SATS extends the perimeter to several hops (multi-hop communication). This can increase significantly the convergence speed compared to single-hop protocols. However, the use of multi-hop communication not only increases connectivity but also increases the end-to-end communication delay. The latter harms synchronization accuracy and consequently affects the consensus convergence. Therefore, each node must carefully select the neighboring nodes to consider for clock synchronization. SATS used a distributed dynamic programming-based approach to select these neighbors. The simulation results show that SATS increase significantly the convergence speed compared to single-hop protocols.

Another alternative to increase the convergence speed of the consensus-based protocol has been presented by Jianping He et al. [46]. The latter proposed to converge the logical clocks to the maximum value among all nodes instead of considering the average. Indeed, the average consensus-based time synchronization algorithms converge to global synchronization asymptotically, while MTS converges to global synchronization in a finite time [46]. This is an interesting property that has been studied in detail in [47]. Two versions of the protocol have been proposed. The maximum time synchronization (MTS), which ignores the communication delays. And the weighted maximum time synchronization (WMTS) consider that the random communication delay follows a normal distribution.

3.2.5 Temperature-Aware Compensation protocols

Temperature variation is the main reason that desynchronizes the clocks in WSANs Indeed, the clocks are equipped with quartz oscillators. The frequency of the oscillator depends on the form of its crystal, which in turn depends on the ambient temperature. The temperature-aware Compensation protocols [60], [61], [62], [48], [37] tries to calculate, according to the ambient temperature, the changes in clocks frequency, and then update the clocks to compensate for this changes. This allows to reduce the clock drift, and prolong the re-synchronization intervals without losing synchronization accuracy.

The Temperature-Compensated Time Synchronization Protocol (TCTS) [60] assumes that a set of sensor nodes in the network (called slave nodes) receive precise timestamps from remote nodes, called master nodes. These slave nodes must compensate for their clock frequency errors caused by temperature change. The TCTS protocol consists of two phases. During the first phase (calibration phase), each slave node learns the relationship between its clock frequency error and the temperature. To do this, the slave node receives calibration messages from the master nodes at regular intervals. Each time the slave node receives a calibration message it records in a regression table its current local time and temperature. During the second phase (Compensation phase), the slave node periodically measures the temperature. Knowing the time offset between the timestamp and local time, the slave node uses linear regression to estimate its current frequency error and compensate its local offset estimation. Other protocols use different estimators to dynamically compensate the clock drift according to the ambient temperature. For example, The temperature-assisted clock self-calibration (TACSC) [37], used the maximum likelihood estimators. And the temperature-compensated Kalman-based distributed synchronization protocol (TKDS) [37] used the Kalman filter. We note that temperature-Aware compensation protocols can be combined with the protocol seen above to extend their re-synchronization period and thus improve their efficiency.

3.2.6 On demand synchronization protocols

In contrast to the proactive protocols seen above, the on-demand (or reactive) synchronization protocols synchronize the node clocks only when an event occurs. This schema makes the on-demand protocols much more efficient in terms of communication and energy consumption but requires a delay to synchronize the node clocks. This delay may constrain some WSANs applications such as detection and tracking systems. In the literature, few time synchronization protocols are reactive [58],[59]. Huang et al. [58] proposed the accurate on-demand time synchronization protocol (AOTSP). The latter assumes that the network forms a breadth-first search from the ROOT node. When an event occurs, the nodes use their local clocks to timestamps this event. Then this timestamp is converted as he is transmitted to the ROOT. To explain this conversion process we assume that a node, which has detected an event, wants to inform the ROOT. This node will then send a message M to the ROOT containing the information concerning the event, and also special fields that allow calculating the elapsed time to transfer the message. These fields are updated by each intermediary node through which the message is transmitted. When the message M reaches the ROOT, it converts the time of the received event to its local time (the global time) by subtracting the transfer time. This process is repeated for each new event. Liua et al. [59] used the same principle as AOTSP and adds a drift compensation mechanism. This allows him to minimize error accumulation during the transfers of the timestamps of the events.

3.3 Time synchronisation protocols in WSANs

In the literature, very few time synchronization protocols have been proposed specifically for WSANs [21, 22]. Martirosyan and BoukercheIn [21] designed the clustered ordering by confirmation protocol (COBC), for preserving temporal relationships of events in WSANs. This protocol is composed of two modules, the first one treats the problem of temporal event ordering, and the second one treats the problem of time synchronization. These problems are linked because to keep the chronological order of events the nodes must be synchronized. By considering temporal event ordering and time synchronization together, the authors want to reduce the protocol's overhead compared to what exists in the literature. To do this, after forming clusters, the cluster heads are synchronized with each other using RBS protocol. Also, the cluster members are synchronized with their corresponding cluster heads using TPSN protocol. To reduce the overhead, COBC combines time synchronization messages with routing and event ordering messages. This preserves the energy of nodes, but can adversely affect time synchronization accuracy. However, if there is not enough activity in the network (routing and event ordering messages) the nodes will not be synchronized correctly.

The approach of considering time synchronization with another related problem has also been employed in [22] and [23]. But this time the authors have integrated the time synchronization with the localization of nodes to minimize the overhead. However, in the protocols using this approach, generally, the time synchronization process can be thwarted, and give less accuracy compared to dedicated protocols.

In addition to the two previous protocols, several time synchronization protocols developed for WSNs can be used for WSANs. However, an important constraint specific to WSANs must be considered. Indeed, in WSANs, the actuators' transmission range is much greater than the one of the sensors. This can generate asymmetric links between actuators and sensors. The sender-receiver protocols, such as TPSN, LTS, TSRT, STETS, CTS, and SPiRT assume that all the links between nodes in the network are symmetric, which is not true in WSANs. However, these protocols can be adapted to WSANs such that the asymmetric links do not negatively influence their time synchronization mechanism. This is also the case for the recipient-recipient protocols, such as RBF and R4Sync, where nodes use symmetric links to exchange time reception of the beacon. In contrast, the sender protocols, such as FTSP, GSTP, FCSA, and RMTS do not require the bidirectional exchange of time information and can be used in the presence of asymmetric links. However, all the protocols designed for WSNs do not consider the additional transmission capacity of actuators and see them as sensors. Of course, this additional capacity can be exploited advantageously to increase time synchronization accuracy.

3.4 Conclusion

In this chapter, we have presented different synchronization protocols for WSNs and WSANs. We note that very few protocols have been proposed for WSANs. However, several protocols designed for WSNs can be used in WSANs, including protocols based on the seder mechanism. Indeed, the latter is not affected by the asymmetric link between sensors and actuators. In the next chapter, we propose a new time synchronization protocol specially designed for WSANs.

Chapter 4

SANSync, An Accurate Time Synchronization Protocol for WSANs

Contents

4.1 Introduction

We propose, in this chapter, a novel time synchronization protocol, namely Sensor and Actuator Networks Synchronization Protocol (SANSync), specifically designed for WSANs. The actuator in SANSync forms clusters. The protocol takes benefits from the large transmission capacity of the actuators to minimize the synchronization error inside the clusters. This significantly increases the accuracy of time synchronization in the whole network. In the following, we describe the idea and the pseudo-code of our protocol and present the simulation results.

4.2 Idea of the proposed protocol

In this section, we describe the idea of the SANSync protocol. We assume that :

- A node in the network can be a sensor or an actuator. The number of sensors is much greater than the number of actuators.
- The transmission range of actuators is greater than the transmission range of sensors.
- Each sensor or actuator node has a unique identifier.

We also assume that a reference node ref has to synchronize the global clocks of all nodes in the network with a reference clock $L_{ref}()$. For that, SANSync forms clusters in the network, each one is composed of : (i) an actuator acting as cluster head, and (ii) a set of sensors and actuators members that are in the transmission range of the cluster head. We note that there are nodes outside the clusters. SANSync use two different mechanisms to synchronize the nodes' clocks, an *extra-cluster* mechanism and an *intra-cluster* mechanism. The first one is used to synchronize the global clocks of nodes outside the clusters. The second is used to synchronize the global clocks of nodes inside the clusters. The intra-cluster mechanism allows reducing time synchronization error of nodes inside clusters. This increases greatly the accuracy of the time synchronization in the whole network. The protocol proceeds as follows :

- *Step 1 (Formation of clusters):* Each actuator node in the network broadcasts a Cluster formation message (CFM) to nodes in its transmission range. A node that receives a CFM from an actuator will consider this latter as its cluster head (if it does not have a cluster head yet), otherwise, the received message will be ignored.

- *Step 2 (Synchronization of the clusters clocks) :* In addition to its physical clock $H_i()$ and its global clock $L_i()$, each node i inside a cluster maintains another logical clock $C_i()$ called *cluster clock*, which will be synchronized with the physical clock of its cluster head. The nodes within clusters, therefore, will have common local time references (the cluster clocks) which will be used in the next step of the protocol to synchronize the global clocks.

- *Step 3 (Synchronization of the global clocks):* In this step, the reference node synchronizes all the global clocks of nodes with a reference clock. The nodes inside clusters are synchronized using the intra-cluster mechanism, the other nodes are synchronized using the extra-cluster mechanism. The main contribution of this paper lies in the use of the local time references (the cluster clocks) to estimate the global clocks in the intra-cluster mechanism, this allows to avoid amplification of time synchronization error inside the clusters and consequently increases the accuracy of time synchronization in the whole network.

To explain the operations of the protocol, we consider, without loss of generality, the WSANs described in figure 4.1. The network is composed of a reference node denoted ref, $n-1$ sensors denoted successively from 1 to $n-1$, and an actuator denoted n.

Each node i can only communicate with its direct neighbors $i-1$ and $i+1$ if any, except for the actuator node which may additionally broadcast messages directly to sensors nodes p to $n-1$, $(1 < p < n-1)$. Therefore, there are asymmetric links between the actuator node n and the sensors nodes p to $n-2$, and symmetric links between adjacent nodes. In the following, we detail the different steps of the protocol.

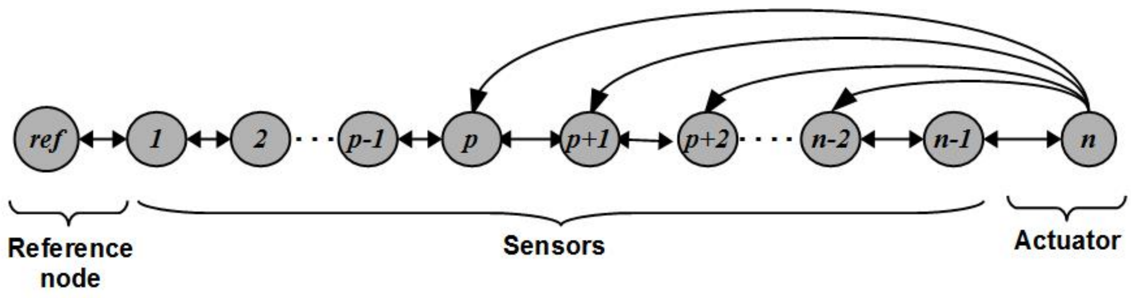


Figure 4.1: A WSANs composed of a reference node denoted ref, $n-1$ sensors denoted successively from 1 to $n-1$ and an actuator denoted n.

4.2.1 Formation of clusters

The actuator node n broadcasts a Cluster formation message (CFM) to all nodes $i, i = p, \cdots n-1$. p is the identifier of the farthest node from actuator n (see figure 4.1). When a node i receives the CFM message from node n, it sets this latter as its cluster head.

4.2.2 Synchronization of the cluster clocks

After forming the cluster, the cluster head n periodically broadcasts its physical clock value to all nodes $i, i = p, \cdots n-1$. Each node $i,\ i = p, \cdots, n-1$, receiving a new value H_n from the cluster head n, records the synchronization point (H_i, H_n) as a pair (x, y) in a least-squares table called*cluster least-squares table (CLST)*, where H_i is the physical clock value of node i when it receives the value H_n from the cluster head n. After recording several points in its CLST, each node $i, i = p,\ \cdots, n-1$, estimates the rate multiplier of its cluster clock $C_i()$ so that it runs at the same speed as the physical clock of the cluster head n.

Assuming that a node i has recorded m points (x, y) in its CLST, the rate multiplier of its logical clock $C_i()$ noted c_i^{head} is calculated as follows:

$$c_i^{head} = \frac{\sum_{j=1}^{m}((x_i^j - \overline{x})(y_i^j - \overline{y}))}{\sum_{j=1}^{m}(x_i^j - \overline{x})^2}, \tag{4.1}$$

where (x_i^j, y_i^j) is the j^{th} point recorded by the node i in its CLST. We note that for a given variable x, $\overline{x}$ represents the empirical mean ($\overline{x} = \frac{1}{n}\sum_{i=1}^{n}(x_i)$).

The rate multiplier of a node i , $i = p, \cdots, n-1$, is updated each time the node adds a new point in its CLST. In cases where there is a single point in the CLST, the rate multiplier is equal to 1. It should be noted that the more points are recorded, the rate estimation is accurate. But since the sensors' storage capacity is limited only the last points are recorded.

After estimating the rate multiplier of its cluster clock, the cluster clock value of a node i at universal time t is calculated as follows :

$$C_i(t) = y_i^{last} + \ c_i^{head}(H_i(t) - x_i^{last}), \tag{4.2}$$

where x_i^{last} and y_i^{last} are the latest values recorded by the node i in its CLST.

Inside the cluster there are no intermediate nodes between the cluster head and the nodes p to $n-1$, the lasts ones will receive the messages from the cluster head approximately at the same time. After calculating the rates multipliers of the cluster clocks, we shall have $C_i(t) \approx C_j(t)$ for each pair i and j of nodes inside the cluster at each instant t.

At the end of this step, the cluster clocks of nodes p to $n-1$ will be synchronized with the cluster-head physical clock and will be used in the next step of the protocol. We note that the least-squares method is used by several protocols in the literature [15, 19,41,63,64,66] and provides good results. Also, a performance analysis aimed at highlighting the sensitivity of this method to estimate the nodes' clocks is given in [41]. However, other methods can be used, such as maximum-likelihood estimators [32,37,67, 69] or nonlinear Gaussian regression synchronization model [68].

4.2.3 Synchronization of the global clocks

In the last steps of the protocol, the reference node synchronizes the global clocks $L_i()$, $i = 1, \cdots, n$, of nodes 1 to n with its reference clock $L_{ref}()$. In doing so, the protocol uses an extra-cluster mechanism to synchronize the nodes outside the cluster (1 to $p-1$), and an intra-cluster mechanism to synchronize the nodes inside the cluster (p to n).

4.2.3.1 Synchronization of nodes 1 to $p-1$ (the extra-cluster mechanism) :

The reference node periodically transmits its reference clock value to node 1. The node 1 receiving a new value L_{ref} from the reference node, records the synchronization point (H_1, L_{ref}) as a pair (x, y) in a least-squares table called *global_least-squares_table (GLST)* . H_1 is the physical clock value of node 1 when it receives the value L_{ref} from the reference node.

After recording enough points in its GLST the node 1 estimates the rate multiplier of its global clock $L_1()$ so that it runs at the same speed as the reference clock $Lref()$. Assuming that the node 1 has recorded m points in its GLST, then the rate multiplier of its global clock $L_1()$ noted l_1^{ref} is calculated as follows:

$$l_1^{ref} = \frac{\sum_{i=1}^{m}((x_i - \overline{x})(y_i - \overline{y}))}{\sum_{i=1}^{m}(x_i - \overline{x})^2}, \tag{4.3}$$

where (x_i, y_i) is the i^{th} point recorded by the node 1 in its GLST.

The rate multiplier of node 1 is updated each time it adds a new point in its GLST. In the cases where there is a single point in the GLST, the rate multiplier is equal to 1.

After estimating the rate multiplier, the global clock value of node 1 at universal time t is calculated as follows :

$$L_1(t) = y^{last} + \ l_1^{ref}(H_1(t) - x^{last}), \tag{4.4}$$

where x^{last} and y^{last} are the latest values recorded by the node 1 in its GLST.

After the node 1 has synchronized its global clock $L_1(t)$, it, in turn, synchronize the global clock of node 2 using the same mechanism, and so on, until the last node $p-1$ is synchronized.

4.2.3.2 Synchronization of nodes p to n (the intra-cluster mechanism) :

After being synchronized, the node $p-1$ periodically transmits the estimation of the reference clock value (its global clock value) to node p. The node p receiving a new value $\widehat{L_{ref}}$ from the node $p-1$, records the synchronization point $(H_p, \widehat{L_{ref}})$ as a pair (x, y) in its *global_least-squares_table (GLST)*. H_p is the physical clock value of node p when it receives the value $\widehat{L_{ref}}$ from the node $p-1$. The node p then transmits values $(C_p, \widehat{L_{ref}})$ to node $p+1$. C_p is the cluster clock value of node p when it receives the value $\widehat{L_{ref}}$ from the node $p-1$. The node $p+1$ in turn transmits values $(C_p, \widehat{L_{ref}})$ to node $p+2$, and so on, until the last node n receives these values.

A node i, $i = p+1, \cdots, n$, which receives a new value $(C_p, \widehat{L_{ref}})$, records the synchronization point $(\widehat{H_i^{old}}, \widehat{L_{ref}})$ as a pair (x, y) in its GLST. $\widehat{H_i^{old}}$ is an estimation of the physical clock value of node i when the node p receives the value $\widehat{L_{ref}}$. It is calculated as follows :

For a node $i, i = p+1, \cdots, n-1$, we have:

$$\widehat{H_i^{old}} = H_i(t) - \left(\frac{C_i(t) - C_p}{c_i^{head}} \right). \tag{4.5}$$

For the cluster head n we have:

$$\begin{aligned}\widehat{H_n^{old}} &= H_n(t) - \left(\frac{C_n(t) - C_p}{c_n^{head}}\right) \\ &= H_n(t) - \left(\frac{H_n(t) - C_p}{1}\right) \\ &\qquad\qquad = C_p.\end{aligned} \tag{4.6}$$

After recording enough points in their GLST each node i, $i = p, \cdots, n$, estimates the rate multiplier of its global clock $L_i()$ so that it run at same speed as the reference clock $Lref()$.

Assuming that a node i has recorded m points in its GLST, then the rate multiplier of its global clock $L_i()$ noted l_i^{ref} is calculated as follows:

$$l_i^{ref} = \frac{\sum_{j=1}^{m}((x_i^j - \overline{x})(y_i^j - \overline{y}))}{\sum_{j=1}^{m}(x_i^j - \overline{x})^2}, \tag{4.7}$$

where (x_i^j, y_i^j) is the j^{th} point recorded by the node i in its GLST.

The global clock value of a node i at universal time t is calculated as follows:

$$L_i(t) = y_i^{last} + \ l_i^{ref}(H_i(t) - x_i^{last}), \tag{4.8}$$

where x_i^{last} and y_i^{last} are the latest values recorded by the node i in its GLST.

At the end of this steps, the global clocks $L_i()$, $i = 1, \cdots, n$, of nodes 1 to n will be synchronized with the reference clock. Existing time synchronization protocols do not take into account the asymmetric links between the actuator node n and the sensors nodes p to $n - 1$. To synchronize global clocks of nodes 1 to n in these protocols, the reference node transmits synchronization messages to node 1. The node 1 synchronizes its global clock $L_1()$ and in turn transmit synchronization messages to node 2, and so on until all global clocks in the network be synchronized. The time synchronization error, in this case, is amplified at each hop. This is essentially due to the message reception delays. In SANSync, synchronization messages also passe through the nodes 1 to n as the existing time synchronization protocols however in our case the time synchronization error is not amplified at the nodes p to n, since the values $(C_p, \widehat{L_{ref}})$ are recorded at the node p, and the nodes $p + 1$ to n will only receive and transmit these values to their neighbors without change. The error caused by message reception delay at nodes $p+1$ to n is thus avoided.

4.3 Pseudo-code of the protocol

In the following, we give the pseudo-code of SANSync. It is assumed that the reference node identifier is $ROOT$, and $Lref()$ is the reference clock. Each node i in the network maintains the following functions and variables:

Functions :

- $H_i()$:it returns the physical clock value of node i.
- $C_i()$: it returns the cluster clock value of node i.
- $L_i()$: it returns the global clock value of node i.

Variables :

- $IdCluster_i$: It is the cluster identifier where the node i is a member. We note that the cluster identifier takes the value of the cluster head identifier. If the node i is not a member of any cluster, then $IdCluster_i$ takes the value $NOCLUSTER$.
- $NodeType_i$: It determines the type of node i (SENSOR or ACTUATOR).
- l_i^{ref} : It is the rate multiplier of node i global clock.
- c_i^{head} : It is the rate multiplier of node i cluster clock.
- seq_i : It is the last sequence number received by the node i.We note that the $ROOT$ node increments its sequence number each time it broadcasts a new reference clock value.
- $GLST_i$: It is the least-squares table used to estimate the global clock of node i.
- $CLST_i$: It is the least-squares table used to estimate the cluster clock of node i.
- $GlobalTimePoint_i$ *and* $ClusterTimePoint_i$: These two variables are used by the node i to record, respectively, the last received reference clock value, and its cluster clock value when it received the reference clock value.

The protocol uses the following three types of messages :

- *ClusterFormation :* This message is used by actuators to form clusters and synchronize the cluster clocks of the nodes inside the clusters. It contains the identifier of the actuator and its physical clock value.
- *ExtraClusterSync :* This message is used by nodes that are outside the clusters to synchronize the global clocks of their neighbors with the reference clock. It contains the node identifier, the last received sequence number, and the estimated value of the reference clock.
- *IntraClusterSync :* This message is used by nodes that are inside the clusters to synchronize the global clocks of their neighbors with the reference clock. It contains the node identifier, the last received sequence number, the estimated value of the reference clock, and the values of the variables *GlobalTimePoint* and *ClusterTimePoint*.

Each actuator or sensor node in the network maintains a timer (*GlobalTimer*) which will fire every $T1$ seconds, it allows the nodes to periodically re-synchronize the global clocks of their neighbor nodes. The actuator nodes maintain additional timers (*ClusterTimer*) which will fire every $T2$ seconds, it allows the actuator nodes to periodically re-synchronize cluster clocks of the nodes inside the clusters.

The pseudo-code of the protocol is as follows:

```
1: **Initialisation of node $i$**
2: $idcluster_i \leftarrow NOCLUSTER$;
3: $seqi \leftarrow 0$;
4: $l_i^{ref} \leftarrow 1$;  $c_i^{head} \leftarrow 1$;
5: $GlobalTimePoint_i \leftarrow 0$;
6: $ClusterTimePoint_i \leftarrow 0$;
7: if ($i = ROOT$) then
8:     Set periodic timer with period $T1$ ($GlobalTimer$);
9: endif;
10: if ($NodeType_i = ACTUATOR$) and ($i \neq ROOT$) then
11:     Set periodic timer with period $T2$ ($ClusterTimer$);
12: endif;
```

```
13: **Upon ClusterTimer of node $i$ time-out**
14: if ($IdCluster_i = NOCLUSTER$ ) then
15:     $IdCluster_i \leftarrow i;$
16: endif;
17: Broadcast $< i, H_i >$ in $ClusterFormation$ message;

18: **Upon node $i$ receiving $CusterFormation$ message $< j, H_j >$ from node $j$**
19: if ( $i \neq ROOT$) then
20:     if ( $IdCluster_i = NOCLUSTER$ ) then
21:         $IdCluster_i \leftarrow j;$
22:     endif;
23:     if ($IdCluster_i = j$) then
24:         $x \leftarrow H_i;$
25:         $y \leftarrow H_j;$
26:         Store $(x, y)$ in $CLST_i$ and estimate $c_i^{head};$
27:     endif;
28: endif;

29:  **Upon $GlobaleTimer$ of node $i$ time-out**
30: if ($i = ROOT$) then
31:     $seq_i \leftarrow seq_i + 1;$
32: endif;
33: if ($IdCluster_i = NOCLUSTER$) then
34: Broadcast $< i, seq_i, L_i >$ in $ExtraClusterSync$ message
35: else
36:     Broadcast $< i, seq_i, IdCluster_i, L_i, GlobalTimePoint_i,$
$ClusterTimePoint_i >$ in $IntraClusterSync$ message;
37: endif;

38: **Upon node $i$ receiving** $ExtraClusterSync$ **message** $< j, seq_j, L_j >$ **from node $j$**
39: if ($seq_j > seq_i$ ) then
40:     $seq_i \leftarrow seq_j;$
41:     if ( $IdCluster_i \neq NOCLUSTER$) then
42:         $GlobalTimePoint_i \leftarrow L_j;$
```

```
43:         $ClusterTimePoint_i \leftarrow C_i$;
44:     endif;
45:     $x \leftarrow H_i$;
46:     $y \leftarrow L_j$;
47:     Store $(x, y)$ in $GLST_i$ and estimate $l_i^{ref}$;
48:   Set periodic timer with period $T1$ ($GlobalTimer$) if it is not started yet;
49: endif;

50: **Upon node *i* receiving *IntraClusterSync* message** $< j, seq_j, IdCluster_j, L_j$
$, GlobalTimePoint_j, ClusterTimePoint_j >$ **from node *j***
51: if ($seq_j > seq_i$ ) then
52:     $seq_i \leftarrow seq_j$;
53:     if ( $IdCluster_i \neq IdCluster_j$ ) then
54:         $x \leftarrow H_i$;
55:         $y \leftarrow L_j$;
56:         if ( $IdCluster_i \neq NOCLUSTER$) then
57:             $GlobalTimePoint_i \leftarrow L_j$;
58:             $ClusterTimePointi \leftarrow C_i$;
59:         endif;
60:     else
61:         $x \leftarrow H_i - ((C_i - ClusterTimePoint_j)/c_i^{head})$;
62:         $y \leftarrow GlobalTimePoint_j$;
63:         $GlobalTimePoint_i \leftarrow GlobalTimePoint_j$;
64:         $ClusterTimePoint_i \leftarrow ClusterTimePoint_j$;
65:     endif;
66:     Store $(x, y)$ in $GLST_i$ and estimate $l_i^{ref}$;
67:   Set periodic timer with period $T1$ ($GlobalTimer$) if it is not started yet;
68: endif;
```

In the beginning, each node initializes its variables (lines 2-6). The ROOT node in addition start its *GlobalTimer* (lines 7-9), and the actuators nodes start their *ClusterTimer* (lines 10-12).

The *ClusterTimer* is used by the actuator nodes to form the clusters and synchronize the cluster clocks of the nodes inside the clusters. The *ROOT* node does not belong to any cluster. Each time the *ClusterTimer* of an actuator node fires, it broadcasts in a *ClusterFormation* message its physical clock value to nodes in its transmission range (line 17). A node that receives a *ClusterFormation* message from an actuator will consider the actuator as its cluster head (if it does not have a cluster head yet), and the identifier of the actuator as its cluster identifier (lines 20-22). Otherwise, this node already belongs to another cluster, or it is the *ROOT* node, the received message will be ignored. Each node that receives a new physical clock value of its cluster head will add a new synchronization point to its *CLST* (lines 23-27). Once there are enough points in its *CLST*, each node inside a cluster can estimate the rate multiplier of its cluster clock using equation 4, and can then calculate the cluster clock value using equation 5.

The *GlobalTimer* is used to synchronize global clocks of all nodes in the network with the reference clock. Each time the *GlobalTimer* of the ROOT node fires, it increments its sequence number (lines 30-32), and broadcasts a new value of the reference clock to its neighbor nodes in a *ExtraClusterSync* message (line 34). Each neighbor node that receives a new value of the reference clock will add a new synchronization point to its *GLST* (lines 45-47). Once there are enough points in its *GLST*, each neighbor node inside a cluster can estimate the rate multiplier of its global clock using equation 8, and can then calculate the global clock value using equation 9. If this node belongs to a cluster, it records the received reference clock values and its cluster clock value, respectively, in the variables *GlobalTimePoint* and *ClusterTimePoint* (lines 41-44). After their logical clocks are synchronized, each neighbor node of the *ROOT* node will in turn synchronizes the global clocks of its neighbors, and so on until all global clocks of nodes in the network will be synchronized.

To synchronize the global clocks of its neighbors, a node uses either an *ExtraClusterSync* message or an *IntraClusterSync* message. If the node does not belong to any cluster, it broadcasts only the estimated reference clock value to its neighbor nodes in a *ExtraClusterSync* message (line 34) which was previously the case of the *ROOT* node. In contrast, if the node belongs to a cluster, it broadcasts in a *IntraClusterSync* message, the estimated reference clock value and also, the values recorded in the variables *GlobalTimePoint* and *ClusterTimePoint* (line 36). A node that receives an *IntraClusterSync* message from a node in its cluster uses the values *GlobalTimePoint* and *ClusterTimePoint* contained in the message to calculate and add a new synchronization point to its *GLST* (lines 60-66), using equation 6. This node also records the two values *GlobalTimePoint* and *ClusterTimePoint* in order to send them later to its neighbors. If the node receiving a *IntraClusterSync* message does not belong to the same cluster as the sender node, it processes the message as a *ExtraClusterSync* message (lines 53-58).

4.4 Simulation

We evaluate the efficiency of SANSync through simulations, as well we compare it with the FTSP and FCSA protocols. FTSP is among the first protocols developed for WSNs and remains a reference to compare new protocols. FCSA is a more recent protocol and remains one of the most performant at the moment. Also, these two protocols can be used, without adaptation, to synchronize node clocks in WSANs.

The purpose of SANSync, FTSP, and FCSA is to allow the different nodes' logical clocks to show relatively close values at any moment. To measure the accuracy of time synchronization, we consider the following metrics :

- *Global skew :* It is the maximum difference between the logical clock value of a node *i* and the logical clock values of the other nodes in the network at a given time instant.
- *Maximum global skew :* It is the maximum of global skews of all nodes at a given time instant.
- *Average global skew :* It is the average of global skews of all nodes at a given time instant.

To the best of our knowledge existing simulators designed for WSNs do not allow the implementation of heterogeneous nodes (sensor and actuator) or have a serious problem of scalability. For these reasons we have developed our simulator based on the C++ language and implemented the three protocols. The variance in the message delay is modeled with a normally distributed random variable. The physical clock drifts were uniformly distributed between -50 ppm and 50 ppm. The re-synchronization period for the three protocols is set to 30 seconds, and the number of entries for the least-squares tables is set to 8. These parameters are the same as those used by the authors of the FCSA protocol in their simulations[19]. Also, the sensor's transmission range is set to 50m, and the actuator's transmission range is set to 200m. In what follows, we will present different simulation scenarios :

Scenario 1 : We first simulated a line topology with 20 nodes from 1 to 20 (see figure 4.2) where the node 10 is an actuator, and the other nodes are sensors. The distance between adjacent nodes is 50 m, and hence each node i can communicate with adjacent nodes $i-1$ and $i+1$ if any, and the actuator can directly send messages to nodes 6 to 9, and nodes 11 to 14. We recall that the actuator transmission range is four-time greater than the transmission range of the sensors. Also, sensor node 1 is the reference node, and its physical clock is the reference clock. We simulated the three protocols during 10 hours, and recorded during the last hour, for each node, the maximum difference observed between its global clock and the reference clock. figure 4.3 shows the synchronization error obtained for each node.

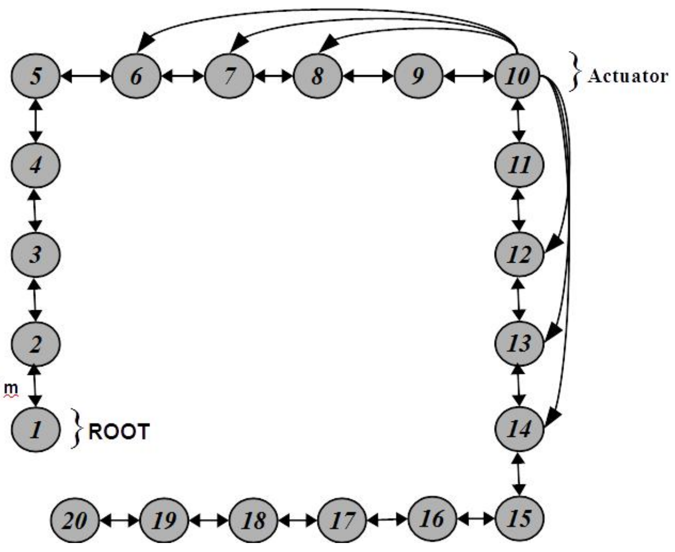


Figure 4.2: A line topology that consists of 20 nodes (19 sensors and an actuator).

From the obtained results, we can see that the synchronization error of nodes 2 to 20 in FTSP and FCSA grows every time we move away from the reference node. This is due to the accumulation of synchronization errors at each hop. These errors are caused essentially by message reception delay. However, we notice that SANSync is distinguished by the fact that the synchronization error does not increase in nodes 7 to 10. This is a significant advantage since it reduces the synchronization error in all the nodes coming after the actuator node 10 compared to FTSP and FCSA. In SANSync, nodes that are inside the same cluster have almost the same synchronization error. In our case, the cluster is formed by nodes 6 to 14, with node 10 as cluster head. We can see that the synchronization error of node 14, which is at the far right of the cluster, is almost equal to the synchronization error of node 6 which is at the far left of the cluster. This is due to the intra-cluster synchronization mechanism of SANSync which allows it to significantly improve the accuracy of clock synchronization inside the clusters.

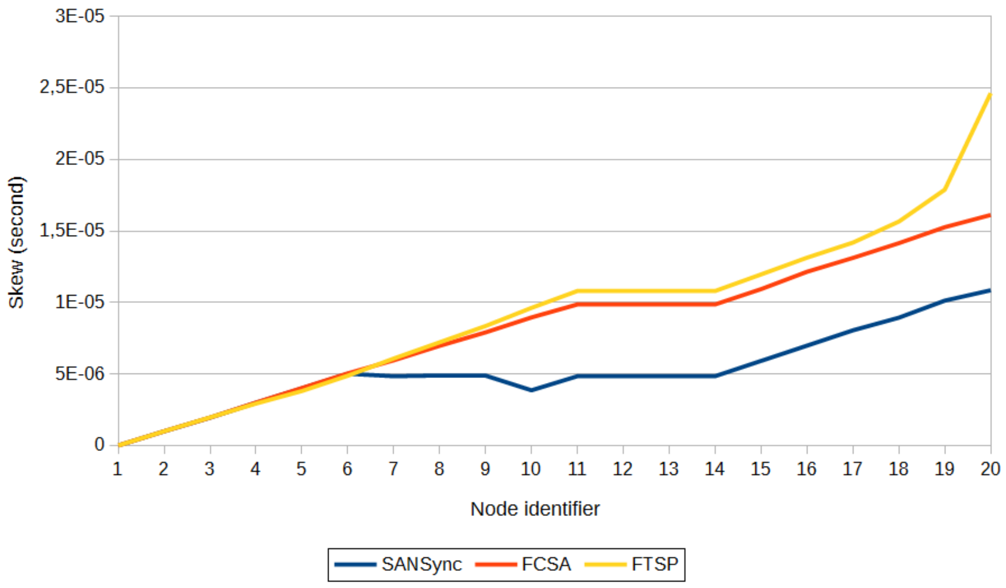


Figure 4.3: Synchronization error of nodes 1 to 20 for SANSync, FTSP and FCSA.

Scenario 2 : To evaluate SANSync performance on a network with a large diameter, we then, simulated a line topology with 50 nodes from 1 to 50 where the nodes 10,20,30,40,50 are actuators, and the other nodes are sensors. Also, sensor node 1 is the reference node, and its physical clock is the reference clock. We simulated the three protocols during 30 hours, and recorded during the last hour, for each node, the maximum difference observed between its global clock and the reference clock. From the obtained results (see figure 4.4 and 4.5), we note that the synchronization error of FTSP grows exponentially with the network diameter, in the most recent FCSA protocol that uses a clock speed agreement method, the synchronization error grows with the square root of the network diameter. Concerning SANSync, we notice that it is more efficient than FCSA and FTSP. In fact, in SANSync, the synchronization error inside the clusters remains stable which allows it to have good results over long distances.

Scenario 3 : To evaluate SANSync performance on a more general network, we then, generated 15 different networks (1 to 15). Each network consists of 20 actuators and

1000 sensors (one actuator for 50 sensors), which are randomly distributed over an area of 1000m x 1000m. The reference node is placed in the center of the simulation area. Each simulation is performed for each network for 48 hours. We recorded, during the last hour, the maximum values observed of the average global skew and the maximum global skew for each network. The obtained results (see figure 4.6, 4.7, 4.8, and 4.9) confirm SANSync performance. SANSync significantly minimizes the time synchronization errors compared to FTSP and FCSA. The average value of the maximum global skew and the average global skew obtained from the 15 different networks are reduced by over 80% compared to FTSP. Also, compared to FCSA, the maximum global skew and the average global skew are reduced by 36% and by 31% respectively.

Scenario 4 : In this last scenario, we tried to vary the number of actuators in the network. For that, we generated 5 different networks (1 to 5). Each network consists of 1000 sensors and several actuators (resp. 5, 10, 20, 40, and 80 actuators), which are randomly distributed over an area of 1000m x 1000m. The reference node is placed in the center of the simulation area. Each simulation is performed for each network for 48 hours. We recorded, during the last hour, the maximum values observed of the average global skew and the maximum global skew for each network. From the obtained results (see figure 4.10, 4.11, 4.12 and 4.13), it can be observed that the performance of the three protocols becomes closely equal by increasing the number of actuators. This can be explained by the fact that the more the number of actuators increases, the more the diameter of the network decreases. In this case, the synchronization errors for the three protocols are not amplified and therefore remain close. Another observation is the fact that SANSync significantly improves the accuracy of the clock synchronization even in cases where there is a limited number of actuators. This is the case for the network with only 5 actuators where the maximum global skew and the average global skew are reduced by over 96% (resp. 62%) compared to FTSP (resp. FCSA).

From the simulations, we can see that SANSync provides a good synchronization quality for small and large-scale WSANs. However, it is more advantageous on networks with large diameters, when the number of actuators and their distribution is appropriate. Nevertheless, since SANSync does not use a clock speed agreement, the synchronization error grows exponentially outside the clusters.

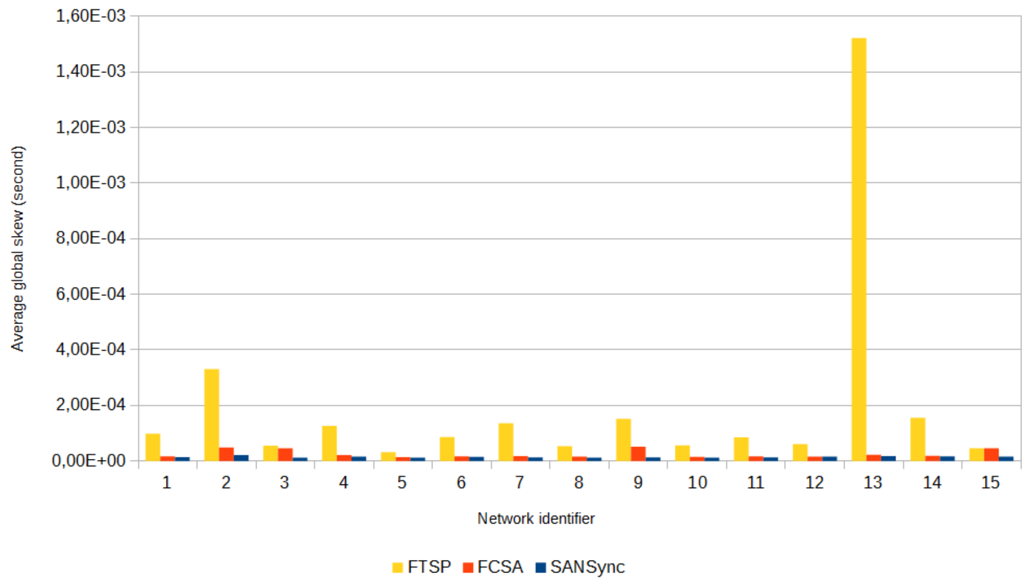


Figure 4.4: The maximum values of the average global skew observed for FTSP, FCSA and SANSync, for networks 1 to 15.

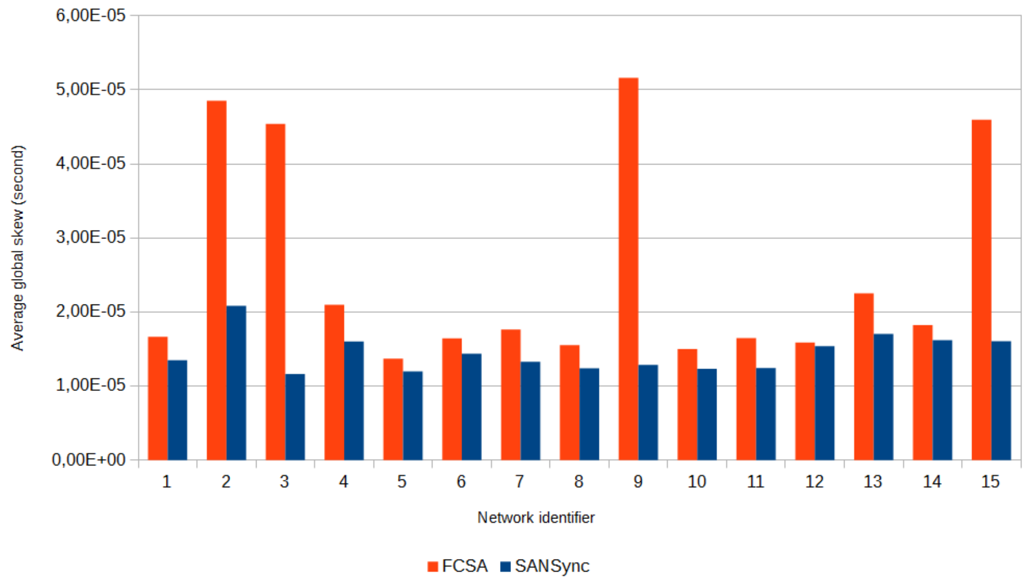


Figure 4.5: The maximum values of the average global skew observed for FCSA and SANSync, for networks 1 to 15.

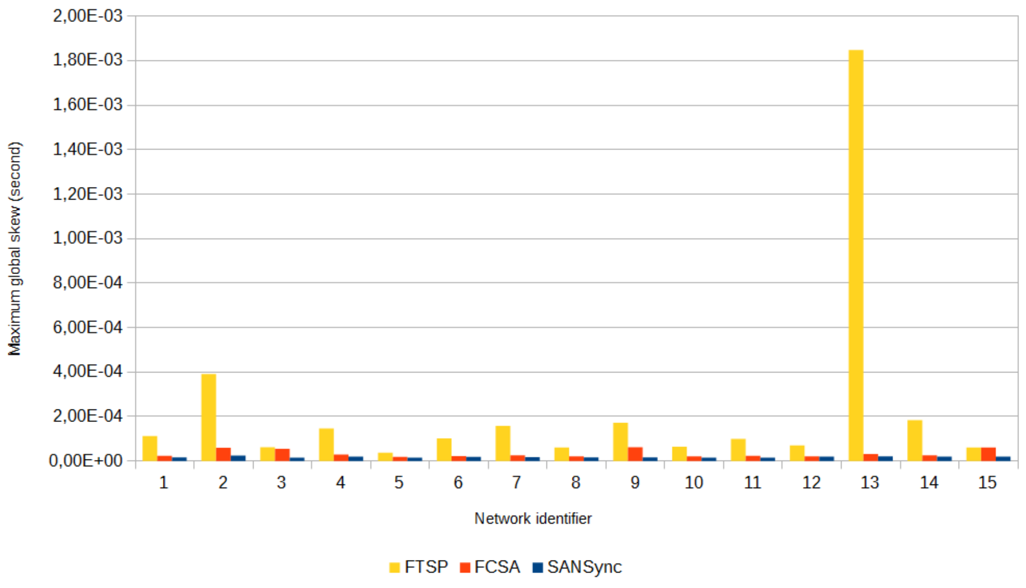


Figure 4.6: The maximum values of the maximum global skew observed for FTSP, FCSA and SANSync, for networks 1 to 15.

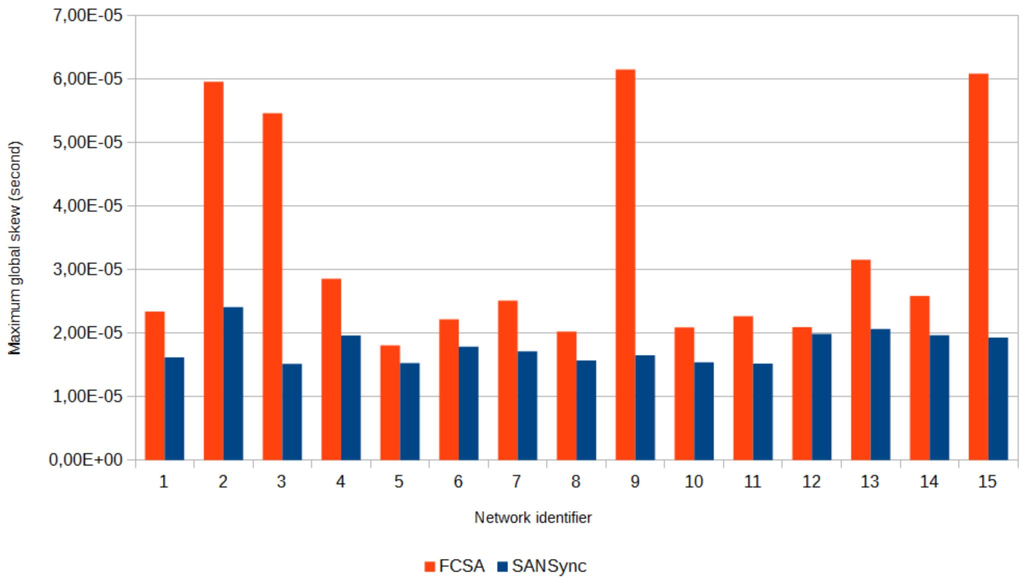


Figure 4.7: The maximum values of the maximum global skew observed for FCSA and SANSync, for networks 1 to 15.

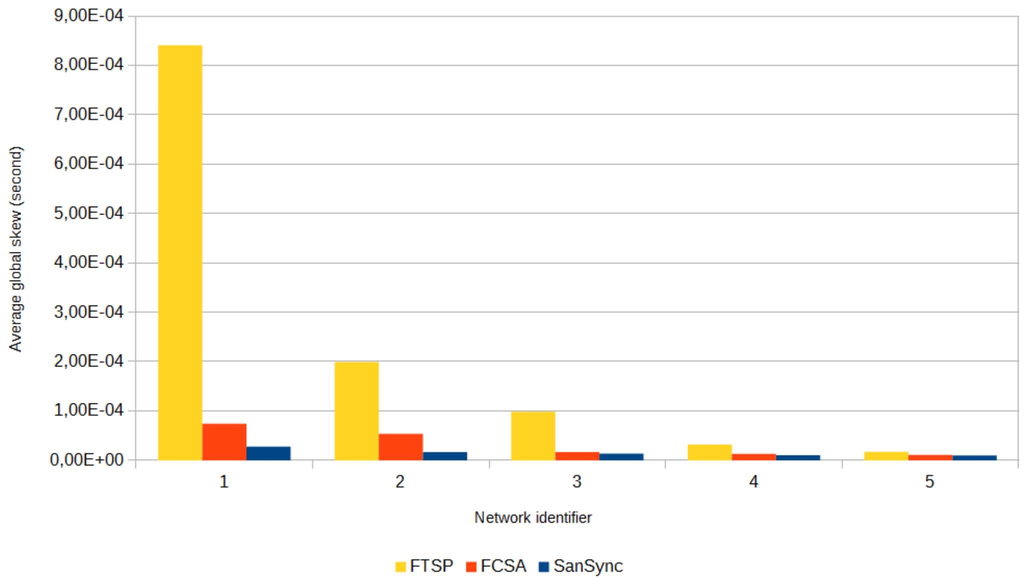


Figure 4.8: The maximum values of the average global skew observed for FTSP, FCSA and SANSync, considering networks : 1 (1000 sensors + 5 actuators), 2 (1000 sensors + 10 actuators), 3 (1000 sensors + 20 actuators), 4 (1000 sensors + 40 actuators), and 5 (1000 sensors + 80 actuators).

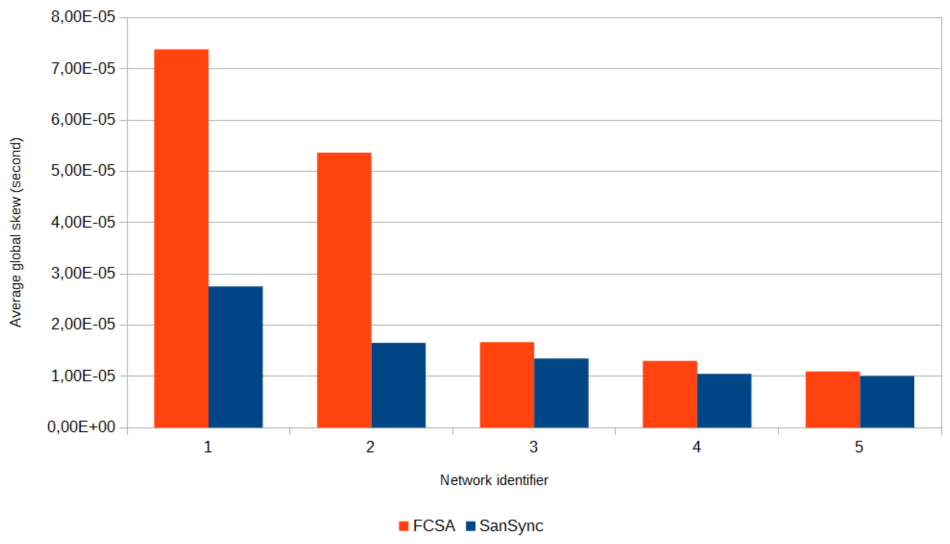


Figure 4.9: The maximum values of the average global skew observed for FCSA and SANSync, considering networks : 1 (1000 sensors + 5 actuators), 2 (1000 sensors + 10 actuators), 3 (1000 sensors + 20 actuators), 4 (1000 sensors + 40 actuators), and 5 (1000 sensors + 80 actuators).

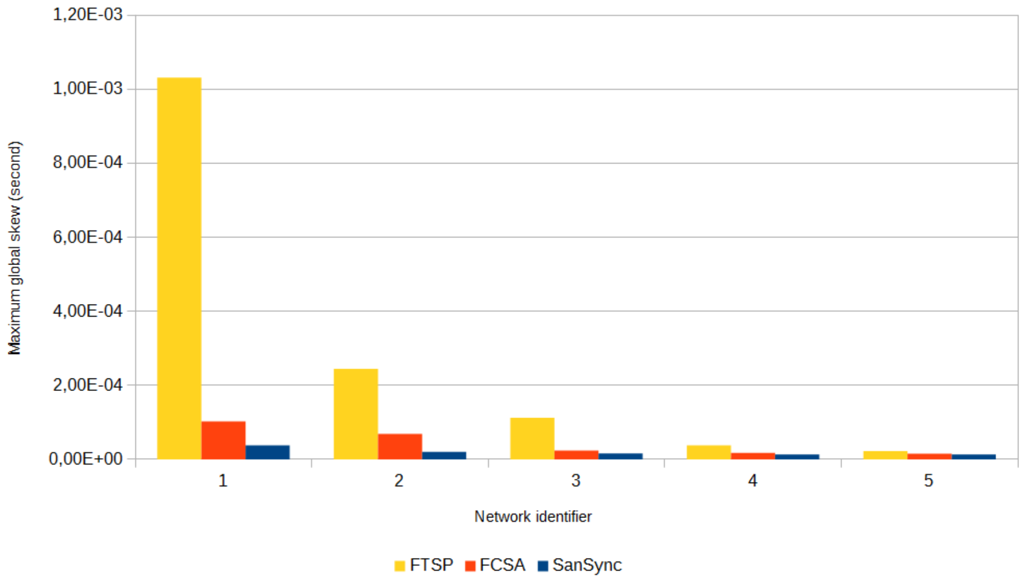


Figure 4.10: The maximum values of the maximum global skew observed for FTSP, FCSA and SANSync, considering networks : 1 (1000 sensors + 5 actuators), 2 (1000 sensors + 10 actuators), 3 (1000 sensors + 20 actuators), 4 (1000 sensors + 40 actuators), and 5 (1000 sensors + 80 actuators).

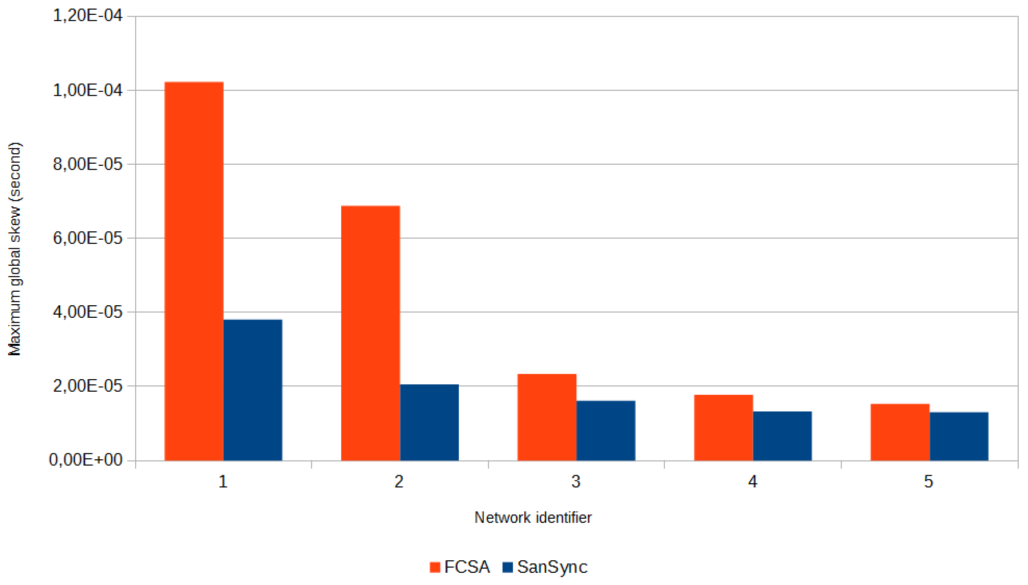


Figure 4.11: The maximum values of the maximum global skew observed for FCSA and SANSync, considering networks : 1 (1000 sensors + 5 actuators), 2 (1000 sensors + 10 actuators), 3 (1000 sensors + 20 actuators), 4 (1000 sensors + 40 actuators), and 5 (1000 sensors + 80 actuators).

4.5 Conclusion

In this chapter, we presented a new time synchronization protocol designed especially for WSANs. Unlike existing time synchronization protocols, the proposed protocol, namely SANSync, exploits the asymmetric links between actuators and sensors to create accurate local time references (cluster clocks). These references were then used to synchronize the global clocks of the nodes. According to the simulation results, SANSync greatly increases the time synchronization accuracy compared with FTSP and FCSA. We note that the performance of SANSync is insured by the intra-cluster mechanism which is used to synchronize nodes inside the clusters.

Chapter 5

An optimized time synchronization protocol by combining SANSync and FCSA protocols

Contents

5.1 Introduction

As seen in Chapter 4, the strong point of SANSync is its inter-cluster synchronization mechanism used to synchronize the nodes inside the clusters. Nevertheless, outside the clusters, the clocks of nodes are synchronized using an extra-cluster mechanism similar to the one used in FTSP. Therefore, the time synchronization error in SANSync grows exponentially outside the clusters. In this chapter, we propose an optimized time synchronization protocol for WSANs. This latter combines the intra-cluster synchronization mechanism of SANSync with a clock speed agreement mechanism used by FCSA. This combination improves the synchronization accuracy of the nodes outside the clusters and gives better results compared to SANSync and FCSA. In the following, we first describe the speed agreement mechanism of FCSA. This later allows the nodes to agree on a common clock speed. Secondly, we provide a detailed pseudo-code of our protocol. Thirdly we propose a multi-ROOTs version of the protocol. Finally, we present the simulation results.

5.2 The clock speed agreement mechanism

In this section, we describe the clock speed agreement mechanism of FCSA. As was said in the introduction, this mechanism allows all nodes in the network to agree on common logical clock speed. This has the effect of reducing the amplification of the time synchronization error. The synchronization error, in this case, grows with the square root of the network diameter [2]. To do this, each node j in the network periodically broadcasts to its neighbor nodes (nodes which are in its transmission range) a synchronization message containing its physical value H_j, its logical clock value L_j, and the rate multiplier l_j. Each time a node i receives a new synchronization message from a neighbor node j, it records the synchronization point (H_i, H_j) as a pair (x, y) in the least-squares table (LST). H_i is the physical clock value of node i when it receives the value H_j from the neighbor node j. We note that there is an LST for each neighbor node. After recording several points in the LST related to the neighbor node j, the node i estimates the relative physical clock rate h_i^j of the neighbor node j using the least-squares regression method as follow :

$$h_i^j = \frac{\sum_{k=1}^{m}((x_i^k - \overline{x})(y_i^k - \overline{y}))}{\sum_{k=1}^{m}(x_i^k - \overline{x})^2}, \tag{5.1}$$

where (x_i^k, y_i^k) is the k^{th} point recorded by the node i in the LST related to the neighbor node j. m is the number of points recorded in the LST.

We note than each node i in the network maintains a repository to keep track of the relative physical clock rates and rate multipliers of its neighbors. These values are then used by the node i to update its rate multiplier li as follow :

$$l_i(t^+) = \frac{l_i(t) + \sum_{j \in N_i} h_i^j(t)\ l_i(t)}{|N_i| + 1}, \tag{5.2}$$

where t^+ represents the time immediately after the update operation. N_i and $|N_i|$

represents the set of neighbor nodes and the number of neighbor nodes of node i, respectively.

With this execution, it can be proven theoretically that all nodes agree on a common logical clock speed $h_i.l_i$ [19].

5.3 An optimised time synchronization protocol

In this section, we propose the Optimized Sensor and Actuator Networks Synchronization Protocol (OSANSync). This latter combines the advantages of SANSync and FCSA protocols. We assume that the actuators nodes have already divided the network into clusters. Each cluster is composed of a cluster head (which is an actuator), and a set of members nodes. We note that there are also nodes outside the clusters.

Each node i in the network maintains the following functions and variables:

- seq_i : is the last sequence number received by the node i. We note that the reference node ($ROOT$) increments its sequence number each time it broadcasts a new reference clock value.
- $IdCluster_i$: is the cluster identifier of the node. We note that the cluster identifier takes value of the cluster head identifier. If the node i is not a member of any cluster, then $IdCluster_i$ takes the value $NOCLUSTER$.
- $L_i()$,$H_i()$ and $C_i()$: are functions that return the logical clock value, the physical clock value, and the cluster clock value, respectively.
- l_i^{ref} and c_i^{head} : are the rate multipliers of the logical clock and the cluster clock, respectively.
- For each neighbor node j, the variable h_i^j stor the physical clock rate of this neighbor.
- $GLST_i$ and $CLST_i$: are the least-squares tables used to estimate the logical clock and the cluster clock, respectively.
- For each neighbor node j, the node i maintains a least-squares table LST_i^j. This table is uses to estimate the physical clock rate h_i^j.
- $LogicalTimePoint_i$ and $ClusterTimePoint_i$: are two variables used to store the synchronization points.

Also, each node in the network maintains a timer T. This latter will fire periodically to re-synchronize the clocks. The pseudo-code of the proposed protocol is as follows :

```
1: **Initialisation of the node $i$**
2: $l_i^{ref} \leftarrow 1$;  $c_i^{head} \leftarrow 1$;
3: For each neighbor node $j$ : $h_i^j \leftarrow 1$;
4: $LogicalTimePoint_i \leftarrow 0$;
5: $ClusterTimePoint_i \leftarrow 0$;
6: $seq_i \leftarrow 0$;
7:     Set periodic timer $T$ with period P;

8: **Upon timer $T$ of the node $i$ time-out**
9: if ($i = ROOT$) then
10:     $seq_i \leftarrow seq_i + 1$;
11: endif;
12: Broadcast to neighbor nodes $< i, IdCluster_i, L_i,$
$H_i, l_i^{ref}, LogicalTimePoint_i, ClusterTimePoint_i, seq_i >$

13: **Upon node $i$ receiving** $< j, IdCluster_j, L_j, H_j, l_j^{ref},$
$LogicalTimePoint_j, ClusterTimePoint_j, seq_j >$ **from node $j$**
14: if ($IdCluster_i = NOCLUSTER$) then
15:    $x \leftarrow H_i$;
16:    $y \leftarrow H_j$;
17:    store $(x, y)$ in $LST_i^j$ and estimate $h_i^j$ using Eq.(5.1);
18:    estimate $l_i^{ref}$ using Eq.(5.2);
19:    if ($seq_j > seq_i$ ) then
20:      $seq_i \leftarrow seq_j$;
21:      $y \leftarrow L_j$;
22:      store $(x, y)$ in $GLST_i$;
23:    endif;
24: else ($IdCluster_i \neq NOCLUSTER$)
25:    if ($IdCluster_i = j$) then
26:    $x \leftarrow H_i$;
27:    $y \leftarrow H_j$;
```

```
28:   store $(x, y)$ in $CLST_i$ and estimate $c_i^{head}$ using Eq.(3.1);
29:   endif;
30:   if ($seq_j > seq_i$ ) then
31:     $seq_i \leftarrow seq_j$;
32:     if ($IdCluster_i = IdCluster_j$) then
33:       $x \leftarrow H_i - ((C_i - ClusterTimePoint_j)/c_i^{head})$Eq.(3.5);
34:       $y \leftarrow LogicalTimePoint_j$;
35:       $LogicalTimePoint_i \leftarrow LogicalTimePoint_j$;
36:       $ClusterTimePoint_i \leftarrow ClusterTimePoint_j$;
37:     else ($IdCluster_i \neq IdCluster_j$)
38:       $x \leftarrow H_i$;
39:       $y \leftarrow L_j$;
40:       $LogicalTimePoint_i \leftarrow L_j$;
41:       $ClusterTimePoint_i \leftarrow C_i$;
42:     endif;
43:     store $(x, y)$ in $GLST_i$ and estimate $l_i^{ref}$
44:     using Eq.(3.7);
45:   endif;
46: endif;
```

In the beginning, each node in the network initializes its logical clock rate multiplier, its cluster clock rate multiplier, and its sequence number (lines 1-7). After that, each node periodically broadcast its synchronization variables to its neighbor nodes (line 12). Thereafter, each node that receives a new synchronization message from a neighbor node, estimate its logical clock rate multiplier using the clock speed agreement mechanism of FCSA (lines 14-23) or the intra-cluster synchronization mechanism of SANSync (lines 24-45). This depends if the node is inside or outside a cluster. Also, each node record a synchronization point in its GLST (lines 22 and 43). We note that a node can at any time calculate its logical clock value using Eq.(3.8).

5.4 Multi-ROOTs synchronization

In WSANs, actuators, more expensive than sensors, can be equipped with a Global Positioning System (GPS). This allows them to synchronize their clocks through satellites.

So we will have several actuators in the network that are synchronized (several ROOTs). The question is how to use these different ROOTs to efficiently synchronize all the node clocks in the network. One approach [35], is to divide the network into several clusters, each cluster is synchronized by one ROOT. A sensor will then join the cluster with the closest ROOT . This approach has several drawbacks. Indeed, the maintenance of the clusters generates an additional load. For example, if a ROOT moves or fails, the entire network has to be reconfigured. In addition, synchronizing the nodes with the nearest ROOT is not always the best solution. Indeed the accuracy depends on the distance between the node and the ROOT but also on the latency of the network. This later is non-deterministic and depends on the network congestion. In the following, we will propose a solution to this problem. Our ideal is to use the propagation speed of the synchronization messages to choose the ROOT instead of using the distances that separate the ROOT from the nodes. In fact, the most accurate time reference for a node is the one that arrives first. In this section, we will modify the previous protocol so that it takes into account several ROOTs simultaneously. In the modified protocol, a node can synchronize with any ROOT according to the propagation speed of the synchronization messages. To do that, in addition to the variables seen previously, each node i in the network maintains the variables $TimeOfDiffusion_i$. This is the initial send time of the latest synchronization message received by node i (time when the ROOT send the synchronization message). The pseudo-code of the proposed becomes as follows (changes are underlined) :

```
1: **Initialisation of the node $i$**
2: $l_i^{ref} \leftarrow 1$;  $c_i^{head} \leftarrow 1$;
3: For each neighbor node $j$ : $h_i^j \leftarrow 1$;
4: $LogicalTimePoint_i \leftarrow 0$;
5: $ClusterTimePoint_i \leftarrow 0$;
6: $\underline{TimeOfDiffusion_i \leftarrow 0;}$
7: Set periodic timer $T$ with period P;

8: **Upon timer $T$ of the node $i$ time-out**
9: if ($i = ROOT$) then
10: $\underline{TimeOfDiffusion_i \leftarrow L_i;}$
11: endif;
```

```
12: Broadcast to neighbor nodes $< i, IdCluster_i, L_i,$
$H_i, l_i^{ref}, LogicalTimePoint_i, ClusterTimePoint_i >$

13: **Upon node *i* receiving** $< j, IdCluster_j, L_j, H_j, l_j^{ref},$
$LogicalTimePoint_j, ClusterTimePoint_j, \underline{TimeOfDiffusion_j} >$ **from node *j***
14: if $(IdCluster_i = NOCLUSTER)$ then
15:    $x \leftarrow H_i;$
16:    $y \leftarrow H_j;$
17:    store $(x, y)$ in $LST_i^j$ and estimate $h_i^j$ using Eq.(5.1);
18:    estimate $l_i^{ref}$ using Eq.(5.2);
19:    if $(TimeOfDiffusion_j > TimeOfDiffusion_i)$ then
20:      $\underline{TimeOfDiffusion_i \leftarrow TimeOfDiffusion_j;}$
21:      $y \leftarrow L_j;$
22:      store $(x, y)$ in $GLST_i;$
23:    endif;
24: else $(IdCluster_i \neq NOCLUSTER)$
25:    if $(IdCluster_i = j)$ then
26:    $x \leftarrow H_i;$
27:    $y \leftarrow H_j;$
28:    store $(x, y)$ in $CLST_i$ and estimate $c_i^{head}$ using Eq.(3.1);
29:    endif;
30:    if $\underline{(TimeOfDiffusion_j > TimeOfDiffusion_i)}$ then
31:      $\underline{TimeOfDiffusion_i \leftarrow TimeOfDiffusion_j;}$
32:      if $(IdCluster_i = IdCluster_j)$ then
33:        $x \leftarrow H_i - ((C_i - ClusterTimePoint_j)/c_i^{head})$ Eq.(3.5);
34:        $y \leftarrow LogicalTimePoint_j;$
35:        $LogicalTimePoint_i \leftarrow LogicalTimePoint_j;$
36:        $ClusterTimePoint_i \leftarrow ClusterTimePoint_j;$
37:      else $(IdCluster_i \neq IdCluster_j)$
38:        $x \leftarrow H_i;$
39:        $y \leftarrow L_j;$
40:        $LogicalTimePoint_i \leftarrow L_j;$
41:        $ClusterTimePoint_i \leftarrow C_i;$
42:      endif;
```

```
43:     store $(x, y)$ in $GLST_i$ and estimate $l_i^{ref}$
44:     using Eq.(3.7);
45:   endif;
46: endif;
```

In the beginning, each node in the network initializes its $TimeOfDiffusion$ (line 6) which allows to identify the priority of the synchronization messages. In fact, a node will only take into account synchronization messages that have a higher $TimeOfDiffusion$ than the previous received message (line 19). After that, each node periodically broadcast its synchronization variables $TimeOfDiffusion_i$ to its neighbor nodes (line 13). Then the neighbors synchronize their clocks and update their $TimeOfDiffusion_i$ (line 30 & 31) We note that the ROOT nodes update the $TimeOfDiffusion$ with their current time before transmitting it (line 10). This modified protocol allow multi-ROOTs synchronization and has a high tolerance to nodes failures. In fact, even if several ROOTs are lost, it continues to synchronize the node clocks.

5.5 Simulation

In this section, we compare OSANSync protocol with SANSync and FCSA. We have developed our own simulator based on the C++ language and implemented the three protocols. The variance in the message delay is modeled with a normally distributed random variable. The physical clocks drifts were uniformly distributed between - 50 ppm and 50 ppm. The sensors transmission range is set to 50 m. The actuators transmission range is set to 200 m. In what follows, we will present two simulation scenarios:

Scenario 1: In the first scenario we simulated a line topology with 70 sensor nodes from 1 to 70 (we note that there are no actuators in the network). The distance between the adjacent node is 50 m, and hence each node i can communicate with adjacent nodes i - 1 and i + 1 if any. The sensor node 1 is the ROOT. We simulated the three protocols during 48 h, and recorded during the last hour, for each node, the maximum difference observed between its logical clock and the ROOT clock. Figure 5.1 shows the synchronization error obtained for each node.

From the obtained results, we note that the synchronization error of SANSync grows exponentially with the network diameter. In FCSA and OSANSync, which use the clock

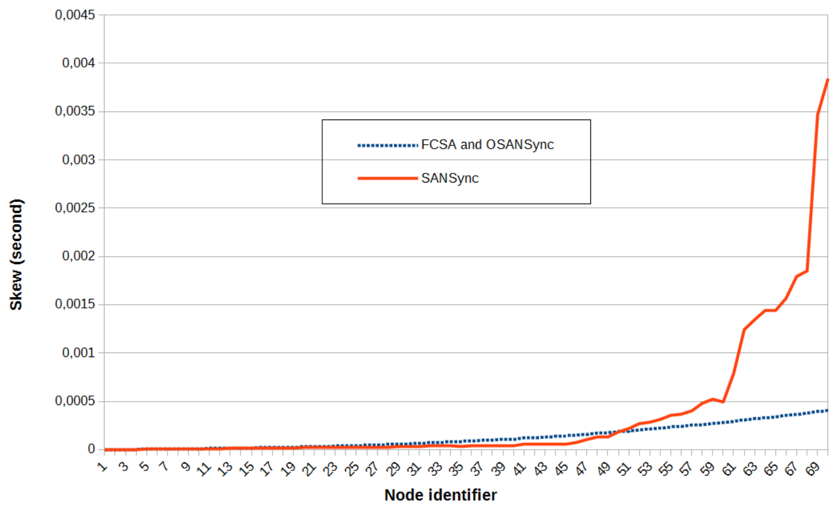


Figure 5.1: Synchronization error of nodes 1 to 70 for SANSync, FCSA and OSANSync.

speed agreement mechanism, the synchronization error grows with the square root of the network diameter.

Scenario 2: In the second scenario we vary the number of actuators in the network. We generated 5 different networks (1-5). Each network consists of 1000 sensors and a number of actuators (resp. 5, 10, 20, 40, and 80 actuators), which are randomly distributed over an area of 1000 m x 1000 m. The *ROOT* is placed in the upper left corner of the simulation area. To measure the accuracy of the time synchronization we considered the maximum global skew and the average global skew.

Each simulation is performed for each network during 48 h. We recorded, during the last hour, the maximum values observed of the average global skew and the maximum global skew for each network. From the obtained results (see figures 5.2 and 5.3), it can be observed that the performance of the three protocols become closely equal by increasing the number of actuators. This is because the more the number of actuators increases, the more the network diameter decreases. In this case, the synchronization error for the three protocols is not amplified and remain close. We also note that OSANSync outperforms FCSA and SANSync in all simulated networks. This is normal since OSANSync combines the advantages of FCSA and SANSync. However, OSANSync greatly improves the time synchronization accuracy when the network diameter is large. For example, in the network 1, the maximum global skew and the average global skew are reduced by over 60% and 35% compared to FCSA, and SANSync, respectively.

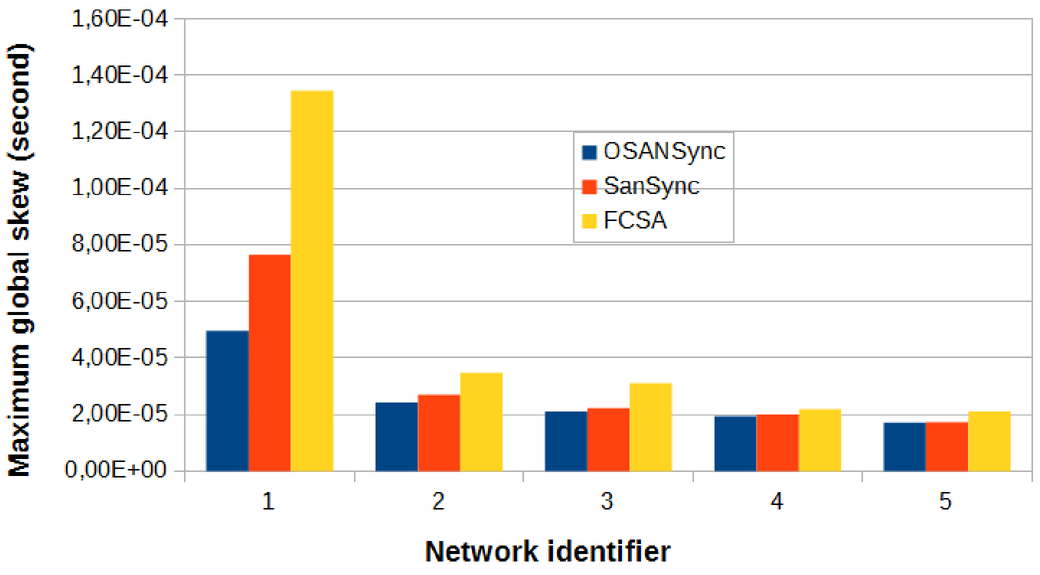


Figure 5.2: The maximum values of the maximum global skew observed for OSANSync,FCSA and SANSync, considering networks: 1 (1000 sensors + 5 actuators), 2 (1000 sensors + 10 actuators), 3 (1000 sensors + 20 actuators),4 (1000 sensors + 40 actuators), and 5 (1000 sensors + 80 actuators).

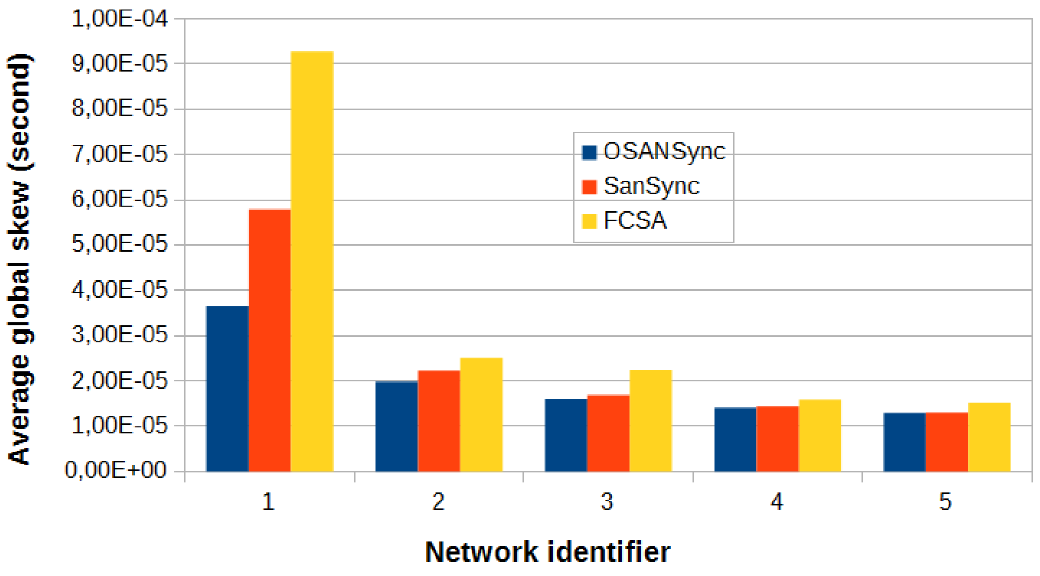


Figure 5.3: The maximum values of the average global skew observed for OSANSync,FCSA and SANSync, considering networks: 1 (1000 sensors + 5 actuators), 2 (1000 sensors + 10 actuators), 3 (1000 sensors + 20 actuators),4 (1000 sensors + 40 actuators), and 5 (1000 sensors + 80 actuators).

5.6 Conclusion

In this chapter, we presented an optimized time synchronization protocol for WSANs. The proposed protocol, namely OSANSync, combines the advantages of FCSA and SANSync protocols. In OSANSync, the intra-cluster synchronization mechanism reduce the error amplification inside the clusters, and the clock speed agreement mechanism reduces the error amplification outside the clusters. This significantly improved time synchronization accuracy. We also provide a multi-ROOTs version of the protocol. This later, which can exploit several ROOTs simultaneously to synchronize the node clocks, can considerably improve time synchronization accuracy. However, it is necessary to implement the multi-ROOTs version of OSANSync in real environments to confirm these assumptions.

Chapter 6

A fully distributed heuristic method for selecting ROOT nodes in WSNs and WSANs

Contents

6.1 Introduction

In the literature, many time synchronization protocols [15, 26, 27, 28, 19, 39, 51, 52] initiate the clocks synchronization process from a given reference node called ROOT. In this case, the synchronization accuracy of a node clock depends on the number of hops that separate it from the ROOT. Indeed, as the number of intermediate nodes increases, the accuracy decreases. This is due to the accumulation of synchronization errors at each hop. For example, in classic FTSP protocol [15], the synchronization error increases exponentially with the network diameter, and in the more recent protocols PulseSync [39] and FCSA [19], the synchronization error increases with the square root of the network diameter. Therefore, the choice of the ROOT is very important since it has a direct influence on time synchronization accuracy. Indeed, choosing a ROOT located in the center of the network will give better results than choosing one in the periphery. This is because, the number of hops between the ROOT and the other nodes is reduced when the ROOT is located at the center of the network. Unfortunately, in the literature, the time synchronization protocols generally do not specify how to select the ROOT or select it without considering the location of the nodes. For example, in FTSP protocol, the ROOT is the node with the lower identifier. In RTSP [51] and RSync [27] protocols, the ROOT is elected according to the energy level of the nodes. In STETS [26] protocol, the ROOT is elected randomly.

In OTSP protocol [52], unlike the previous protocols, the ROOT is selected according to the location of nodes in the networks. Indeed, the ROOT is selected to minimize the maximum distance (number of hops) between the ROOT and any other node in the network. To do this OTSP proceeds as follows:

1. All nodes in the network run a discovery phase to identify their neighbors.
2. After the discovery phase, each node sends the list of its neighbors to the base station.
3. Based on the information received from all the sensor nodes, The base station builds the topology of the entire network and selects the ROOT. This latter is the node with the highest number of neighbors and the lowest number of hops from the base station.

The OTSP protocol selects the ROOT intelligently according to the network topology. However, to do this, OTSP requires the base station to know the complete topology of the network. This centralized approach is not suitable for large distributed networks with limited resources like WSNs. In this chapter, we propose a distributed heuristic method to select ROOT in WSNs. The objective is to minimize the average distance and the maximum distance between the ROOT and all other nodes in the network. The proposed method is fully distributed and may be integrated into different time synchronization protocols to improve their performance. According to the simulation results, our method allows, on average, to select the ROOT among the 5% best nodes in the network that can be set as ROOT.

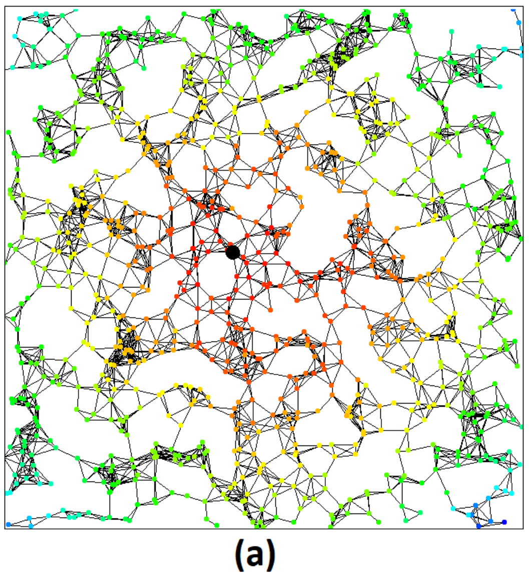
6.2 The best ROOT node for time synchronization

A WSN is modeled by a random geometric graph $G = (V; E)$ [10], where the set $V = \{i : i = 1, ..., N\}$ represents sensor nodes, and the edges set $E = \{(i, j) : i, j \in V\}$ represents bidirectional communication channels. The neighbors of a node $i \in V$, denoted $Ni = \{j \in V : (i, j) \in E\}$, are the set of nodes in the transmission range of i.

As seen above, the position of a node relative to the other nodes is an important parameter for selecting the ROOT. Several metrics exist in the literature to characterize the position of the nodes. For example, the *degree centrality* of a node i is calculated based on the number of its neighbors, the *betweenness centrality* is calculated based on the number of distinct paths in the network that include the node i. The *closeness centrality* is calculated based on the distance of node i from all other nodes in the network. The algorithms used to calculate these metrics are generally centralized and are not adapted to a distributed and resource-limited system such as WSNs. In the following, we will define a list of these metrics [54, 55, 56, 57] :

- The *distance* between any nodes i and j in G , denoted by $d(i, j)$, is the number of edges on the shortest path between i and j.
- The *eccentricity* $e(i)$ of a node i in G , is the maximal distance between the node i and the other nodes in G. The eccentricity of a node i is calculated as follows :
$$e(i) = max\{\, d(i, j) \,|\, j \in V\}. \tag{6.1}$$
- The *closeness centrality* measures how close a node is to all other nodes in the network. The closeness centrality of a node i in G, noted $C(i)$, is calculated as follows :
$$C(i) = \frac{1}{\sum\{d(i, j) \,|\, j \in V\}}. \tag{6.2}$$

To measure the accuracy of time synchronization in WSNs, we have considered maximum global skew and the average global skew. The choice of the best ROOT node for time synchronization depends on the location of the nodes and the objective of the synchronization protocol. Indeed, if the objective is to minimize the maximum global skew, then the best node to choose is the one with the minimum eccentricity. On the other hand, if the objective is to minimize the average global skew, then the best choice, in this case, is the node with the maximal closeness centrality. In Figure 1, we show the eccentricity (Figure 6.1a) and the closeness centrality (Figure 6.1b) of each node in a WSN. In (Figure 6.1a) the red nodes have the lowest values, and blue nodes have the highest. In (Figure 6.1b) the blue nodes have the lowest values, and red nodes have the highest. The black node in the center has the minimum value among all the other nodes. We can notice that the best choice of the ROOT can differ depending on the metric used.



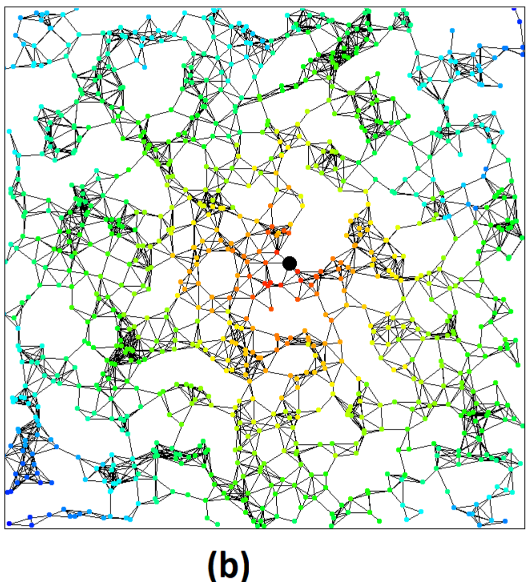


Figure 6.1: The eccentricity (Figure 6.1a), and the closeness centrality (Figure 6.1b) of each node in a WSN .

6.3 The proposed ROOT Selection Method

In this section, we describe a heuristic-based method to select the ROOT node in WSNs. This heuristic is itself based on a distributed average consensus algorithm. The consensus algorithms allow a set of nodes in a network that initially measures some scalar values to know the average of measurements in a distributed way [80]. In our case, the scalars are generated randomly. The average consensus algorithm is iterative, it proceeds as follows:

1) Initially, each node i in the WSN generates randomly a consensus number $x_i(0)$.

2) Periodically, each node of the network broadcasts its consensus number to its neighbors.

3) A node i which receives all the consensus numbers $x_j(t)|\{j \in N_i\}$ from its neighbors, updates its consensus number $x_i(t+1)$ as follows :

$$x_i(t+1) = \frac{x_i(t) + \sum\{x_j(t)|j \in N_i\}}{|N_i| + 1}, \tag{6.3}$$

$|N_i|$ is the number of neighbors of node i.

4) After several iterations, the consensus numbers of all nodes in the network converge to a single consensus value x^* where :

$$x^* = \frac{\sum\{x_i(0)|i \in V\}}{|V|}, \tag{6.4}$$

$|V|$ is the number of nodes in the network. The Proofs of convergence of average consensus algorithms similar to this algorithm have been detailed in [19, 40, 53].

We simulated this algorithm several times for different networks to calculate the average of the consensus numbers generated initially. During these simulations, we noticed that, for the nodes located in the center of the network, the values of the consensus numbers are relatively stable and changes slowly compared to the other nodes. So, our idea is to exploit this characteristic to select the ROOT for time synchronization. Indeed, after several iterations, our method chooses as ROOT the node with the most stable consensus number (which changes the least). To measure the stability of a consensus number of a node i, we calculate at each iteration the difference (in percent) between the new value of the consensus number $x_i(t+1)$ and the old value $x_i(t)$. Figure 6.2 shows, after 1000 iterations, the percentage of change of the consensus numbers in a WSN composed of 1000 nodes. The red nodes have the lowest percentages, and the blue nodes have the highest percentages. The black node has the minimum percentage of change among all the other nodes. Therefore, this latter is selected as ROOT.

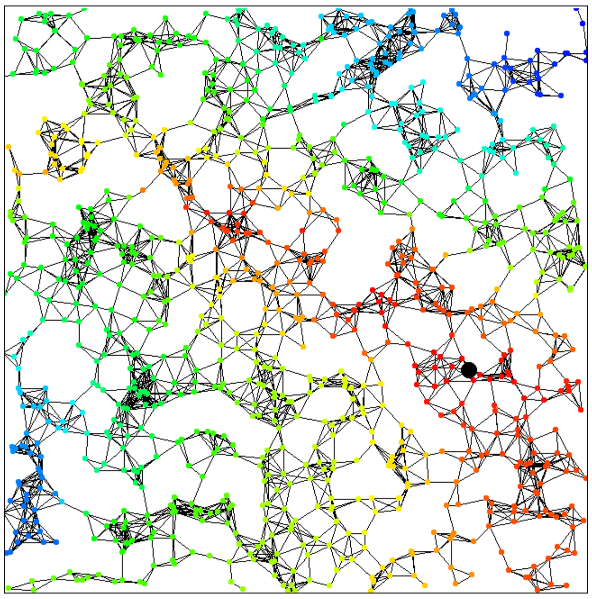


Figure 6.2: The percentage of change of the consensus values in a WSN composed of 1000 nodes.

From Figure 6.2, we can note that the selected ROOT is not located in the center of the network. Indeed, if we use only one initial consensus number for each node, the results are not satisfactory. However, by generating several initial consensus numbers, we will get better results. The algorithm in this case becomes as follows:

1) Each node i generate randomly an initial consensus vector U_i containing n consensus numbers $\{x_i^k | k = 1, \cdots, n\}$.

2) Periodically, each node in the network broadcasts its consensus vector to its neighbors.

3) A node i which receives all the consensus vectors $U_j(t) | \{j \in N_i\}$ from its neighbors, updates each consensus numbers $\{x_i^k | k = 1, \cdots, n\}$ in its consensus vector $U_i(t+1)$ as follows :

$$x_i^k(t+1) = \frac{x_i^k(t) + \sum\{x_j^k(t) | j \in N_i\}}{|N_i| + 1}. \tag{6.5}$$

4) After several iterations, the initial consensus vectors of all nodes in the network converge to a single consensus vector U^*.

To choose the ROOT, we average, for each node i, the percentage of change of each consensus number $x_i^k | k = 1, \cdots, n$ in the consensus vector V_i. We select as ROOT the node that has the vector with the minimum average (which changes the least). Figure 6.3 shows the results obtained from the same network as in Figure 6.2, but this time, each node initially generates a vector containing three consensus numbers. We can see that the selected ROOT is located in the center of the network. It is also shown in Figure 6.4 that the percentage of change of the consensus vector of a node is globally correlated to its closeness centrality and eccentricity. Indeed, the pearson correlation coefficients [81] for closeness centrality and eccentricity are $+0,85$ and $-0,89$ respectively.

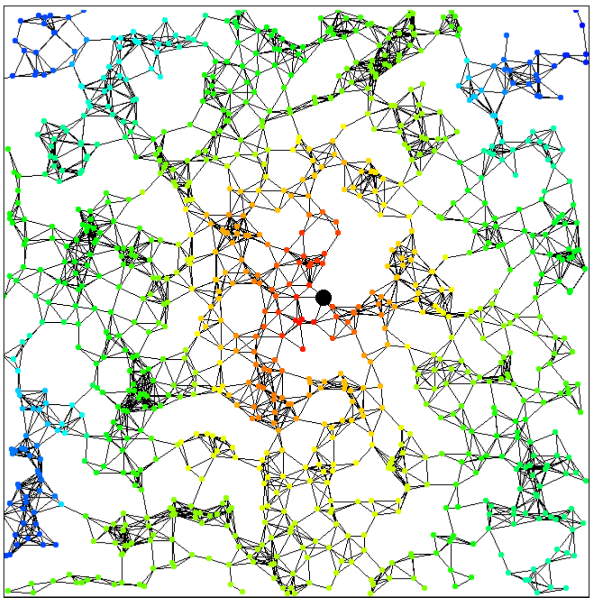
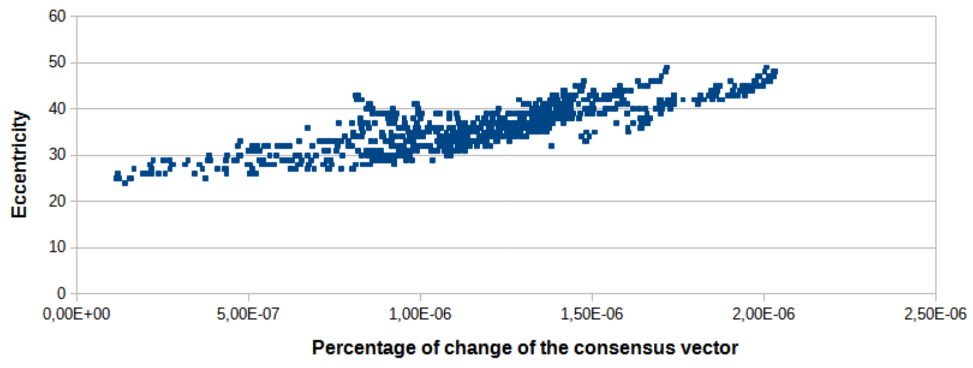


Figure 6.3: The percentage of change of the consensus vectors in a WSN composed of 1000 nodes.

After several iterations, each node i records the percentage of change of its consensus vector in a variable denoted M_i and begins the final phase of ROOT selection. This phase is described by the following algorithm :



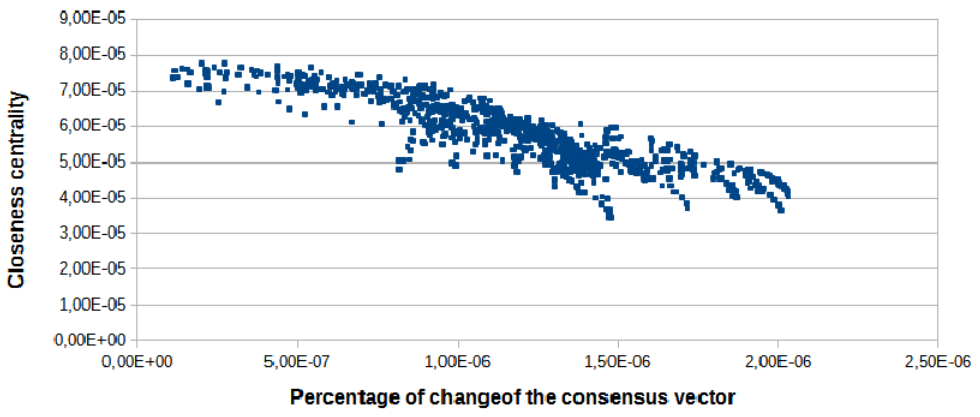


Figure 6.4: Correlation between the ROOT selection metric and the node eccentricity.

```
1: **Initialisation of node $i$**
2:   $ROOT_i \leftarrow i$;
3:   $M_i^{root} \leftarrow M_i$;
4:   Set timer with period $T$;

5: **Upon the timer of node $i$ time-out**
6:   Broadcast to neighbors $< M_i^{root}, ROOT_i >$;

7: **Upon node $i$ receiving values** $< M_j^{root}, ROOT_j >$ **from node $j$**
8:   if ($M_j^{root} < M_i^{root}$) then
9:       $M_i^{root} \leftarrow M_j^{root}$;
10:      $ROOT_i \leftarrow ROOT_j$;
11:      Set timer with period $T$;
12:  endif;
```

$ROOT_i$ is the identifier of the ROOT selected by node i, and the M_i^{root} is the percentage of change of its consensus vector. In the beginning, each node i considers itself as ROOT. So, $ROOT_i$ is set to the node identifier i, and M_i^{root} is set to M_i (lines 2 and 3). Afterward, these two values are broadcasted to the neighbors (line 6). Each node j that receives the values $ROOT_i$ and M_i^{root} from the node i chooses as ROOT the node with the lowest M^{root} value (lines 7 and 10). If the $ROOT$ of node j changes, it arms the timer to broadcast the new information to its neighbors (line 11). After a while, all nodes in the network select as ROOT the node with the lowest M value.

6.4 Simulation

6.4.1 Case of WSNs networks

To evaluate our method we generated using C++ language 1000 different networks. Each network consists of 1000 sensor nodes randomly distributed over an area of 1000 m x 1000 m. The sensors' transmission range is 50 m. We compared, for each network, the selected ROOT node with the other nodes according to its closeness centrality and eccentricity. We have varied the size of the consensus vectors (table I and II) and the number of iterations (table III and VI). The obtained results are described in the following:

Table 6.1: Closeness centrality

Number of iterations	**Size of consensus vector**	**Better**$^{(1)}$	**Worst**$^{(2)}$	**Equivalent**$^{(3)}$
1000	3	11.2859	88.6041	0.11
1000	5	6.5234	93.3664	0.1102
1000	10	4.8551	95.0351	0.1098

$^{(1)}$Percentage of nodes with a closeness centrality less than the ROOT eccentricity (average for 1000 networks), $^{(2)}$Percentage of nodes with a closeness centrality greater than ROOT eccentricity (average for 1000 networks), $^{(3)}$ Percentage of nodes with a closeness centrality equal to the ROOT eccentricity (average for 1000 networks).

We can see from the obtained results that our ROOT selection method, with a sufficient number of iterations, gives very satisfactory results. Indeed, with 1000 iterations, and ten initial consensus numbers, our method selects the ROOT among the 5% best-positioned nodes in the network (nodes with minimum closeness centrality and eccentricity). We can improve these results by increasing the size of the consensus vector.

Table 6.2: Eccentricity

Number of iterations	**Size of consensus vector**	**Better**[(1)]	**Worst**[(2)]	**Equivalent**[(3)]
1000	3	11.3052	85.6108	3.084
1000	5	6.4954	90.9183	2.5863
1000	10	4.632	93.0308	2.3372

[(1)]Percentage of nodes with a eccentricity less than the ROOT eccentricity (average for 1000 networks), [2)]Percentage of nodes with a eccentricity greater than ROOT eccentricity (average for 1000 networks), [(3)] Percentage of nodes with a eccentricity equal to the ROOT eccentricity (average for 1000 networks).

Table 6.3: Closeness centrality

Number of iterations	**Size of consensus vector**	**Better**[(1)]	**Worst**[(2)]	**Equivalent**[(3)]
250	10	29.121	70.7665	0.1121
500	10	14.0616	85.8286	0.1098
1000	10	4.8551	95.0351	0.1098
2000	10	4.8237	95.0659	0.1104

[(1)]Percentage of nodes with a closeness centrality less than the ROOT closeness centrality (average for 1000 networks), [2)]Percentage of nodes with a closeness centrality greater than ROOT v (average for 1000 networks), [(3)] Percentage of nodes with a closeness centrality equal to the ROOT closeness centrality (average for 1000 networks).

Table 6.4: Eccentricity

Number of iterations	**Size of consensus vector**	**Better**[(1)]	**Worst**[(2)]	**Equivalent**[(3)]
250	10	31.1857	63.5675	5.2468
500	10	15.0211	81.094	3.8849
1000	10	4.632	93.0308	2.3372
2000	10	4.0312	93.7851	2.1837

[(1)]Percentage of nodes with a eccentricity less than the ROOT eccentricity (average for 1000 networks), [2)]Percentage of nodes with a eccentricity greater than ROOT eccentricity (average for 1000 networks), [(3)] Percentage of nodes with a eccentricity equal to the ROOT eccentricity (average for 1000 networks).

However, the number of iterations needed to select the ROOT depends on the number of nodes and the depth of the network. Indeed, large networks with a higher number of nodes require more iterations than small ones.

6.4.2 Case of WSANs networks

To evaluate our ROOT selection method in WSANs, we generated 1000 different networks. Each network consists of 1000 sensors and 20 actuators randomly distributed over an area of 1000 m x 1000 m. The sensors' transmission range is 50 m, and the actuators' transmission range is 200 m. We note that the WSANs are modeled as a directed graph. We compared, for each network, the selected ROOT node with the other nodes in the network according to its closeness centrality and eccentricity. We have set the size of the consensus vectors to 10 and the number of iterations to 2000. The obtained results are described in Figure 5. We show that our method also provides good results for WSANs. Indeed, in this configuration, the ROOT is chosen among the 5% best-positioned nodes in the network (nodes with maximal closeness centrality and minimal eccentricity). This result is similar to that of WSNs. We conclude that the presence of asymmetric links does not negatively affect the ROOT selection method.

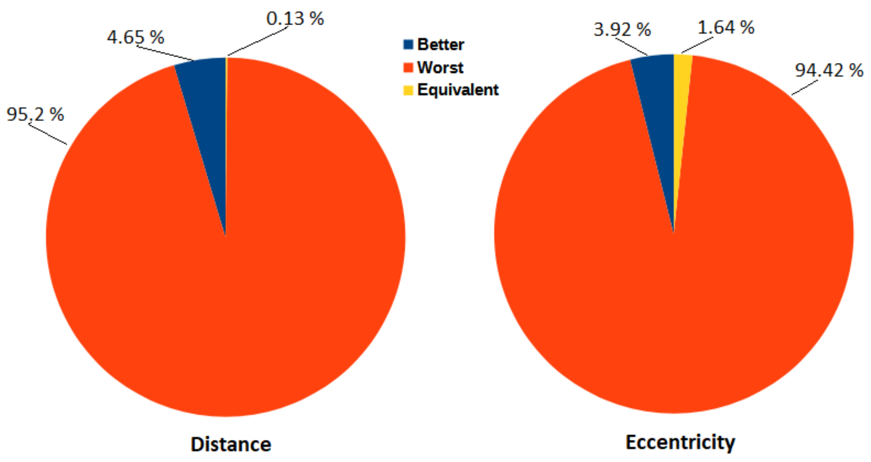


Figure 6.5: ROOT selection in WSANs according to the closeness centrality and eccentricity

We notice that actuators can also have better quality clocks than sensors. For these reasons, it may be judicious to choose the ROOT from the actuators. Indeed, having a more stable reference clock improves synchronization accuracy. Our ROOT selection method allows selecting the ROOT node from all the nodes in WSANs, or only from the actuators. It depends on the chosen strategy.

6.5 Conclusion

In this chapter, we proposed a new ROOT selection method for time synchronization protocols. We can integrate this method into existing or future time synchronization protocols to choose the best ROOT node. In the simulated networks, we successfully selected the ROOT from the 5% best-positioned nodes. The proposed method is fully distributed and robust to node failures and topology changes. It is also possible to increase the size of the consensus vectors and the number of iterations depending on the network size and topology. The more you increase these two parameters the better the results are, but also the network load increases.

Chapter 7

Conclusions and perspectives

In this thesis, we considered the time synchronization problem in WSANs. First, we have thoroughly studied the time synchronization protocols proposed in the literature. The main difficulty of these protocols is how to avoid or compensate for the synchronization message delays. Indeed, these delays are often non-deterministic and cumulative. Then, we proposed a new time synchronization protocol specifically designed for WSANs. Our protocol, called SANsync, exploits the large transmission range of the actuators to reduce the negative impact of the intermediate nodes used in the synchronization process. This strategy has significantly improved the accuracy of time synchronization. Afterward, we combined SANSync with the FTSP protocol, which allowed us to reduce the error amplification, especially when the network diameter is large. And as a result, we have improved the accuracy of the synchronization throughout the network. However, the resulting protocol requires additional resources to store information from neighboring nodes. In the end, we were interested in how to choose the ROOT node used to synchronize all the other nodes of the network. This choice is very important since it directly influences the performance of the synchronization protocols. Indeed, the synchronization accuracy of a node depends on its distance from the ROOT. Surprisingly, this question has not been much discussed in the literature. Thus, we proposed a fully distributed method to solve this problem. Our method can be integrated into the synchronization protocols to choose the ROOT from the best-positioned nodes in the network.

The works presented in this thesis are in progress and can be improved. As perspectives, it will be interesting to work on the following issues:

- Integrate a temperature compensation algorithm into the proposed protocols. This will extend their re-synchronization period and thus improve their efficiency.
- Use the quadratic approach to model clocks. Of course, this approach requires more computation but is more accurate than the linear one.
- Better manage node mobility, in particular the ability to quickly reconfigure clusters.

We also believe that the proposed ROOT selection method can be used to solve other problems such as routing or load balancing. Indeed, it gives an important centrality metric that can be easily exploited by different protocols.

Bibliography

[1] Mohd Fauzi Othmana, Khairunnisa Shazali. Wireless Sensor Network Applications: A Study in Environment Monitoring System. *Procedia Engineering, Volume 41, 2012, Pages 1204-1210, (2012).*
https://doi.org/10.1016/j.proeng.2012.07.302

[2] lAqeel-ur-Rehmanab, Abu Zafar Abbasib ,Noman Islamb, Zubair Ahmed Shaikhb. A review of wireless sensors and networks' applications in agriculture. *Computer Standards & Interfaces Volume 36, Issue 2, February 2014, Pages 263-270. (2014).*
https://doi.org/10.1016/j.csi.2011.03.004

[3] Mohamed Rawidean Mohd Kassim, Ibrahim Mat, Ahmad Nizar Harun. Wireless Sensor Network in precision agriculture application. *2014 International Conference on Computer, Information and Telecommunication Systems (CITS), (2014).*
https://doi.org/10.1109/CITS.2014.6878963

[4] A.Belghith S.Obaidat. Chapter 2 - Wireless sensor networks applications to smart homes and cities. *Smart Cities and Homes Key Enabling Technologies 2016, Pages 17-40, (2016).*
https://doi.org/10.1016/B978-0-12-803454-5.00002-X

[5] Ali JameelAL-Mousawi, Haider. K. AL-Hassani. A survey in wireless sensor network for explosives detection. *Computers & Electrical Engineering Volume 72, Pages 682-701, (2018).*
https://doi.org/10.1016/j.compeleceng.2017.11.013

[6] Natthapol Watthanawisuth, Adisorn Tuantranont, Teerakiat Kerdcharoen. Design for the next generation of wireless sensor networks in battlefield based on ZigBee.

2011 Defense Science Research Conference and Expo (DSR), (2011).
https://doi.org/10.1109/DSR.2011.6026825

[7] Pawel Kulakowski, Eusebi Calle, Jose L Marzo. Sensors-actuators cooperation in WSANs for fire-fighting applications. *2010 IEEE 6th International Conference on Wireless and Mobile Computing, Networking and Communications, (2010).*
https://doi.org/10.1109/WIMOB.2010.5644860

[8] Hamra Afzaal; Nazir Ahmad Zafar. Robot-based forest fire detection and extinguishing model. *2016 2nd International Conference on Robotics and Artificial Intelligence (ICRAI), (2016).*
https://doi.org/10.1109/ICRAI.2016.7791238

[9] Venkata Reddy Palleti, Varghese Kurian, Shankar Narasimhan, Raghunathan Rengaswamy. Actuator network design to mitigate contamination effects in Water Distribution Networks. *Computers and Chemical Engineering 108 194-205,(2018).*
http://dx.doi.org/10.1016/j.compchemeng.2017.09.003

[10] Hichem Kenniche, Vlady Ravelomananana . Random Geometric Graphs as model of Wireless Sensor Networks.*2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE), Singapore, (2010).*
https://doi.org/10.1109/ICCAE.2010.5451758

[11] MIAO XU,WENYUAN XU,TINGRUI HAN, ZHIYUN LIN. Energy-Efficient Time Synchronization in Wireless Sensor Networks via Temperature-Aware Compensation. *ACM Transactions on Sensor Networks, Vol. 12, No. 2, Article 12, (2016).*
http://dx.doi.org/10.1145/2876508

[12] Miao Xu, Wenyuan Xu, Tingrui Han, Zhiyun Lin. Energy-Efficient Time Synchronization in Wireless Sensor Networks via Temperature-Aware Compensation. *ACM Transactions on Sensor Networks No.: 12, (2016).*
https://doi.org/10.1145/2876508

[13] Thomas Schmid, Zainul Charbiwala, Roy Shea, Mani B. Srivastava. Temperature Compensated Time Synchronization. *IEEE Embedded Systems Letters, Volume: 1, Issue: 2, (2009).*
https://doi.org/10.1109/LES.2009.2028103

[14] Cheng-Yu Han. Clock Synchronization and Localization for Wireless Sensor Network. *Thesis, Universite Paris-Saclay, (2018).*
https://tel.archives-ouvertes.fr/tel-01959241

[15] Maroti, M., Kusy, B., Simon, G., Ledeczim, A. The flooding time synchronization protocol.*In SenSys 04 proceedings of the 2nd international conference on embedded networked sensor systems, (2004).*
https://doi.org/10.1145/1031495.1031501

[16] van Greunen, J., Rabaey, J. Lightweight time synchronization for sensor networks. *In WSNA 03: Proceedings of the 2nd ACM international conference on wireless sensor networks and applications, (2003).*
https ://doi.org/10.1145/941350.941353

[17] Swain, A. R., Hansdah, R. C. A model for the classification and survey of clock synchronization protocols in WSNs. *Ad Hoc Networks Volume 27, Pages 219-241, (2015).*
https://doi.org/10.1016/j.adhoc.2014.11.021

[18] Yik-Chung Wu, Qasim Chaudhari, Erchin Serpedin. Clock synchronization of wireless sensor networks. *IEEE Signal Processing Magazine, 28(1), 124-138, (2011).*
https://doi.org/10.1109/MSP.2010.938757

[19] Kasim Sinan Yildirim, Aylin Kantarci. Time Synchronization Based on Slow-Flooding in Wireless Sensor Networks. *IEEE Transactions on Parallel and Distributed Systems, Volume: 25, Issue: 1, (2014).*
https://doi.org/10.1109/TPDS.2013.40

[20] YongGang Miao, XianZhong Tian, TongSen HU, BoJie Fan, Jian Pan, Wei Xu. The Application of Mathematics in Time Synchronization Algorithm for Wireless Sensor Networks. *2009 ISECS International Colloquium on Computing, Communication, Control, and Management, (2009).*
https://doi.org/10.1109/CCCM.2009.5267610

[21] Anahit Martirosyan, Azzedine Boukerche. Preserving Temporal Relationships of Events for Wireless Sensor Actor Networks. *IEEE Transactions on Computers, Volume: 61, Issue: 8, (2012).*
https://doi.org/10.1109/TC.2011.215

[22] Jingfang Li, Cailian Chen and Xinping Guan. XY-expansion: Joint search and clock synchronization for wireless sensor and actor networks. *11th IEEE International Conference on Control & Automation (ICCA) June 18-20, Taichung, Taiwan, (2014). https://10.1109/ICCA.2014.6871034*

[23] Jun Liu, Zhaohui Wang, Jun-Hong Cui, Shengli Zhou,Bo Yang. A Joint Time Synchronization and Localization Design for Mobile Underwater Sensor Networks. *IEEE Transactions on Mobile Computing, Volume: 15, Issue: 3, (2016). https://doi.org/10.1109/TMC.2015.2410777*

[24] Fengyuan Gong; Mihail L. Sichitiu. On the Accuracy of Pairwise Time Synchronization . *IEEE Transactions on Wireless Communications, Volume: 16, Issue: 4, (2017). https://doi.org/10.1109/TWC.2017.2671862*

[25] Zhehan Ding, N. Yamauchi. An improvement of energy efficient multi-hop time synchronization algorithm in wireless sensor network. *IEEE International Conference on Wireless Communications, Networking and Information Security, (2010). https://doi.org/10.1109/WCINS.2010.5541751*

[26] Tie Qiu, Lin Chi, Weidong Guo, Yushuang Zhang. STETS: A novel energy-efficient time synchronization scheme based on embedded networking devices. *Microprocessors and Microsystems Volume 39, Issue 8, November 2015, Pages 1285-1295, (2015). https://doi.org/10.1016/j.micpro.2015.07.006*

[27] Tie Qiu, Yushuang Zhang, Daji Qiao, Xiaoyun Zhang, Mathew L. Wymore, Arun Kumar Sangaiah. A Robust Time Synchronization Scheme for Industrial Internet of Things. *IEEE Transactions on Industrial Informatics Volume: 14, Issue: 8, (2018). https://doi.org/10.1109/TII.2017.2738842*

[28] Sepideh Nazemi GelyanArash Nasiri EghbaliLaleh RoustapoorSeyed Amir Yahyavi Firouz AbadiMehdi Dehghan. SLTP: Scalable Lightweight Time Synchronization Protocol for Wireless Sensor Network. *Mobile Ad-Hoc and Sensor Networks, Third International Conference, MSN 2007, Beijing, China, December 12-14, Proceedings, (2007). https://10.1007/978-3-540-77024-4_49*

[29] Masoume Jabbarifar, Alireza Shameli Sendi, Hosein Pedram, Mahdi Dehghan, Michel Dagenais. L-SYNC: Larger Degree Clustering Based Time-Synchronisation for

Wireless Sensor Network. *2010 Eighth ACIS International Conference on Software Engineering Research, Management and Applications, (2010).*
https://doi.org/10.1109/SERA.2010.30

[30] Jian Zhang, Shiping Lin, Dandan Liu. Cluster-Based Time Synchronization Protocol for Wireless Sensor Networks. *International Conference on Algorithms and Architectures for Parallel Processing ICA3PP 2014: Algorithms and Architectures for Parallel Processing pp 700-711. (2014).*
https://doi.org/10.1007/978-3-319-11197-1_55

[31] Barnabas C. Okeke K. L. Eddie Law, K. L. Multi-level clustering architecture and protocol designs for wireless sensor networks. *WICON '08: Proceedings of the 4th Annual International Conference on Wireless Internet, (2008).*

[32] Chafika Benzaid, Miloud Bagaa, Mohamed Younis. Efficient clock synchronization for clustered wireless sensor networks. *Ad Hoc Networks Volume 56, Pages 13-27, (2017).*
https://doi.org/10.1016/j.adhoc.2016.11.003

[33] G. S. S. Chalapathi, Vinay Chamola, S. Gurunarayanan, Biplab Sikdar. E-SATS: An Efficient and Simple Time Synchronization Protocol for Cluster- Based Wireless Sensor Networks. *IEEE Sensors Journal Volume: 19, Issue: 21, Nov.1, 1, (2019).*
https://doi.org/10.1109/JSEN.2019.2922366

[34] Ganeriwal, S., Kumar, R., Srivastava, M. B. Timing-sync protocol for sensor networks. *Proceedings of the 1st international conference on embedded networked sensor systems, (2003).*
https://doi.org/10.1145/958491.958508

[35] Surendra Rahamatkar, Ajay Agarwal. A Reference Based, Tree Structured Time Synchronization Approach and its Analysis in WSN. *International Journal of Ad hoc Sensor & Ubiquitous Computing abs/1103.4905(1), (2011).*
http://dx.doi.org/10.5121/ijasuc.2011.2103

[36] Jeremy Elson, Lewis Girod and Deborah Estrin. Fine-grained network time synchronization using reference broadcasts. *In Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI), Boston, (2002).*

[37] Zhe Yang, Xi'an, Liang He, Lin Cai, Jianping Pan. Temperature-Assisted Clock Synchronization and Self-Calibration for Sensor Networks, IEEE Transactions on Wireless

Communications. *IEEE Transactions on Wireless Communications Volume: 13, Issue: 6, (2014).*
https://doi.org/10.1109/TWC.2014.051414.130270

[38] Huang, G., Zomaya, A. Y., Delicato, F. C., Pires, P. F. Long term and large scale time synchronization in wireless sensor networks. *Computer Communications Volume 37, 1 January 2014, Pages 77-91, (2014).*
https://doi.org/10.1016/j.comcom.2013.10.003

[39] Christoph Lenzen, Philipp Sommer, Roger Wattenhofer. PulseSync: An Efficient and Scalable Clock Synchronization Protocol. *IEEE/ACM Transactions on Networking Volume: 23, Issue: 3, (2015).*
https://doi.org/10.1109/TNET.2014.2309805

[40] Philipp Sommer, Roger Wattenhofer. Gradient clock synchronization in wireless sensor networks. *2009 International Conference on Information Processing in Sensor Networks, (2009).*
https://doi.org/10.1145/1602165.1602171

[41] Kasim Sinan Yildirim, Aylin Kantarci. External Gradient Time Synchronization in Wireless Sensor Networks. *IEEE Transactions on Parallel and Distributed Systems Volume: 25, Issue: 3, (2014).*
https://doi.org/10.1109/TPDS.2013.58

[42] Luca Schenatoa, Federico Fiorentin. Average TimeSynch: A consensus-based protocol for clock synchronization in wireless sensor networks. *Automatica Volume 47, Issue 9, Pages 1878-1886, (2011).*
https://doi.org/10.1016/j.automatica.2011.06.012

[43] Jie Wu, Liyi Zhang, Yu Bai, and Yunshan Sun. Cluster-Based Consensus Time Synchronization for Wireless Sensor Networks. *IEEE Sensors Journal Volume: 15, Issue: 3, (2015).*
https://doi.org/10.1109/JSEN.2014.2363471

[44] Niranjan Panigrahi, Pabitra Mohan Khilar. Multi-hop consensus time synchronization algorithm for sparse wireless sensor network: A distributed constraint-based dynamic programming approach. *Ad Hoc Networks Volume 61, Pages 124-138, (2017).*
https://doi.org/10.1016/j.adhoc.2017.04.002

[45] Francesco Lamonaca, Andrea Gasparri, Emanuele Garone, Domenico Grimaldi. Clock Synchronization in Wireless Sensor Network With Selective Convergence Rate for Event Driven Measurement Applications. *IEEE Transactions on Instrumentation and Measurement Volume: 63, Issue: 9, (2014). https://doi.org/10.1109/TIM.2014.2304867*

[46] Jianping He, Peng Cheng, Ling Shi, Jiming Chen, and Youxian Sun. Time synchronization in WSNs: A maximum value based consensus approach. *2011 50th IEEE Conference on Decision and Control and European Control Conference, (2011). https://doi.org/10.1109/CDC.2011.6161443*

[47] Guodong Shi and Karl Henrik Johansson. Convergence of distributed averaging and maximizing algorithms Part II: State-dependent graphs. *2013 American Control Conference, (2013). https://doi.org/10.1109/ACC.2013.6580916*

[48] Fengyuan Gong, Mihail L. Sichitiu. Temperature compensated Kalman distributed clock synchronization. *Ad Hoc Networks Volume 62, Pages 88-100, (2017). https://doi.org/10.1016/j.adhoc.2017.04.009*

[49] Kan Xie, Qianqian Cai, Minyue Fu. A fast clock synchronization algorithm for wireless sensor networks. *Automatica Volume 92, June 2018, Pages 133-142 , (2018). https://doi.org/10.1016/j.automatica.2018.03.004*

[50] Fanrong Shi, Xianguo Tuo , Simon X. Yang , Jing Lu, and Huailiang Li. Rapid-Flooding Time Synchronization for Large-Scale Wireless Sensor Networks. *IEEE Transactions on Industrial Informatics, Volume: 16, Issue: 3, (2020). https://doi.org/10.1109/TII.2019.2927292*

[51] Muhammad Akhlaq ; Tarek R. Sheltami. RTSP: An Accurate and Energy-Efficient Protocol for Clock Synchronization in WSNs. *IEEE Transactions on Instrumentation and Measurement, Volume: 62, Issue: 3, (2013). https://doi.org/10.1109/TIM.2012.2232472*

[52] Rachid Beghdad, Faouzi Tinsalhi. OTSP: an optimised time synchronisation protocol for wireless sensor networks. *International Journal of Sensor Networks (IJSNET), Vol. 19, No. 3/4, (2015). https://doi.org/10.1504/IJSNET.2015.072867*

[53] E.Lovisari, S.Zampieri. Performance metrics in the average consensus problem: A tutorial. *Annual Reviews in Control Volume 36, Issue 1, April 2012, Pages 26-41, (2012). https://doi.org/10.1016/j.arcontrol.2012.03.003*

[54] G.Chartrand, S.Tian. Distance in Digraphs. *Computers & Mathematics with Applications Volume 34, Issue 11, December 1997, Pages 15-23, (1997). https://doi.org/10.1016/S0898-1221(97)00216-2*

[55] Matjaz Krnc. Centrality Measures of Large Networks. *Doctoral thesis , University of Primorska, (2015).*

[56] Linton C. Freeman. A Set of Measures of Centrality Based on Betweenness. *Sociometry Vol. 40, No. 1, pp. 35-41, (1977). https://doi.org/10.2307/3033543*

[57] Stephen P.Borgattia, Martin G.Everet. A Graph-theoretic perspective on centrality. *Social Networks, Volume 28, Issue 4, Pages 466-484, (2006). https://doi.org/10.1016/j.socnet.2005.11.005*

[58] Ge Huang, Albert Y.Zomaya, Flavia C.Delicato, Paulo F.Pires. An accurate on-demand time synchronization protocol for wireless sensor networks. *Elsevier Journal of Parallel and Distributed Computing Volume 72, Issue 10, Pages 1332-1346 , (2012). https://doi.org/10.1016/j.jpdc.2012.06.003*

[59] Longgeng Liua, Guangchun Luoa, Ke Qina, Xiping Zhangb . An on-demand global time synchronization based on data analysis for wireless sensor networks. *Procedia Computer Science, Volume 129, Pages 503-510, (2018). https://doi.org/10.1016/j.procs.2018.03.031*

[60] Thomas Schmid, Zainul Charbiwala, Roy Shea, and Mani B. Srivastava. Temperature Compensated Time Synchronization. *IEEE Embedded Systems Letters ,Volume: 1, Issue: 2, (2009). https://doi.org/10.1109/LES.2009.2028103*

[61] MIAO XU, WENYUAN XU, TINGRUI HAN , ZHIYUN LIN. Energy-Efficient Time Synchronization in Wireless Sensor Networks via Temperature-Aware Compensation Share on. *ACM Transactions on Sensor NetworksApril 2016 Article No.:12, (2016). https://doi.org/10.1145/2876508*

[62] Wasan Lasoi and Sataporn Pornpromlikit. Temperature-aware Time Synchronization with an Accuracy-efficiency Trade-off in Wireless Sensor Networks. *Procedia Engineering Volume 168, 2016, Pages 1706-1709, (2016).*
https://doi.org/10.1016/j.proeng.2016.11.495

[63] Christoph Lenzen, Philipp Sommer,Roger P Wattenhofer. Optimal clock synchronization in networks. *SenSys '09: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems, (2009).*
https://doi.org/10.1145/1644038.1644061

[64] AndrÃ© C. Pinho; Daniel R. Figueiredo; Felipe M. G. FranÃ§a. A robust gradient clock synchronization algorithm for wireless sensor networks. *2012 Fourth International Conference on Communication Systems and Networks , (2012).*
https://doi.org/10.1109/COMSNETS.2012.6151304

[65] Domenico Capriglione, Deborah Casinelli, Luigi Ferrigno. Analysis of quantities influencing the performance of time synchronization based on linear regression in low cost WSNs. *Measurement, Volume 77, Pages 105-116, (2016).*
https://doi.org/10.1016/j.measurement.2015.08.039

[66] Juan J.PÃ©rez-Solano, Santiago Felici-Castell. Adaptive time window linear regression algorithm for accurate time synchronization in wireless sensor networks. *Ad Hoc Networks Volume 24, Part A, January 2015, Pages 92-108 , (2015).*
https://doi.org/10.1016/j.adhoc.2014.08.002

[67] Heng Wang, Lun Shao, Min Li, Baoguo Wang, Ping Wang. Estimation of Clock Skew for Time Synchronization Based on Two-Way Message Exchange Mechanism in Industrial Wireless Sensor Networks. *IEEE Transactions on Industrial Informatics ,Volume: 14, Issue: 11, (2018).*
https://doi.org/10.1109/TII.2018.2799595

[68] Divya Upadhyay,Ashwani Kumar Dubey, P.Santhi Thilagam. Application of Non-Linear Gaussian Regression-Based Adaptive Clock Synchronization Technique for Wireless Sensor Network in Agriculture. *IEEE Sensors Journal, Volume:18, Issue: 10, (2018).*
https://doi.org/10.1109/JSEN.2018.2818302

[69] Djamel Djenouri. R4Syn : Relative Referenceless Receiver/Receiver Time Synchronization in Wireless Sensor Networks. *IEEE Signal Processing Letters , Volume: 19,*

Issue: 4, (2012).
https://doi.org/10.1109/LSP.2012.2185491

[70] Weidong Guo, Tie Qiu, Lei Wang, Diansong Luo, Jie Liu. A New Energy-efficient Time Synchronization Algorithm. *2nd International Conference on Information Technology and Electronic Commerce (ICITEC 2014), (2014).*
https://doi.org/10.1109/ICITEC.2014.7105566

[71] Yuan Yongqiong, Zhou Feng, Fu Yuhui, Li Junlu. Simulation of time synchronization protocol of underwater network with acoustic communication cluster topology. *IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC), (2017).*
https://doi.org/10.1109/ICSPCC.2017.8242597

[72] Heng Wang, Daijin Xiong, Liuqing Chen, and Ping Wang. A Consensus-Based Time Synchronization Scheme With Low Overhead for Clustered Wireless Sensor Networks. *IEEE Signal Processing Letters , Volume: 25, Issue: 8, (2018).*
https://doi.org/10.1109/LSP.2018.2847231

[73] Parminder Kaur, Abhilasha. An Energy Efficient Time Synchronization Protocol for Wireless Sensor Networks using Clustering. *2015 IEEE Power, Communication and Information Technology Conference (PCITC), (2015).*
https://doi.org/10.1109/PCITC.2015.7438081

[74] Xiangli Jia 1, Yang Lu, Xing Wei, Wenjing Tao. Improved Time Synchronization Algorithm for Wireless Sensor Networks based on Clustering . *2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC), (2019).*
https://doi.org/10.1109/ITAIC.2019.8785426

[75] Fanrong Shi, Xianguo Tuo, Lili Ran, Zhenwen Ren, Simon X. Yang. Fast Convergence Time Synchronization in Wireless Sensor Networks Based on Average Consensus. *IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, VOL. 16, NO. 2, (2020).*
https://doi.org/10.1109/TII.2019.2936518

[76] Jianping He, Xiaoming Duan, Peng Cheng, Ling Shi, Lin Cai. Accurate clock synchronization in wireless sensor networks with bounded noise. *Automatica Volume 81, Pages 350-358, (2017).*
https://doi.org/10.1016/j.automatica.2017.03.009

[77] Sakshi Rana, Poonam Saini. Consensus time synchronization with clustering in wireless sensor networks. *8th International Conference on Computing, Communication and Networking Technologies (ICCCNT), (2017).*
https://doi.org/10.1109/ICCCNT.2017.8204108

[78] Yu-Ping Tian. Time Synchronization in WSNs With Random Bounded Communication Delays. *IEEE Transactions on Automatic Control , Volume: 62, Issue: 10, (2017).*
https://doi.org/10.1109/TAC.2017.2697683

[79] Kasim Sinan Yildirim, Onder Gurcan. Efficient Time Synchronization in a Wireless Sensor Network by Adaptive Value Tracking. *IEEE Transactions on Wireless Communications, Volume: 13, Issue: 7, (2014).*
https://doi.org/10.1109/TWC.2014.2316168

[80] Sateeshkrishna Dhuli, Y. N. Singh. Analysis of Average Consensus Algorithm for Asymmetric Regular Networks. *CoRR abs/1806.03932, (2018).*

[81] Natarajan Meghanathan. Correlation Coefficient Analysis of Centrality Metrics for Complex Network Graphs. *Intelligent Systems in Cybernetics and Automation Theory (pp.11-20), (2015).*
https://doi.org/10.1007/978-3-319-18503-3_2

[82] Ian F. Akyildiz, Ismail H. Kasimoglu. Wireless sensor and actor networks: research challenges. *Ad Hoc Networks, Volume 2, Issue 4, Pages 351-367, (2004).*
https://doi.org/10.1016/j.adhoc.2004.04.003

[83] Djamel Djenouri. R4Syn : Relative Referenceless Receiver/Receiver Time Synchronization in Wireless Sensor Networks. *IEEE Signal Processing Letters, Volume: 19, Issue: 4, Pages: 175-178, (2012).*
https://doi.org/10.1109/LSP.2012.2185491

[84] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM Volume: 21, Issue 7, (1978).*
https://doi.org/10.1145/359545.359563

[85] Colin J. Fidge. Timestamps in message-passing systems that preserve partial ordering. *Proc. 11th Austral. Comput. Sci. Conf. (ACSC88), (1988).*

Abstract

The purpose of time synchronization is to allow the different nodes' clocks in a network to get relatively close values at any moment. Currently, time synchronization is a fundamental problem in wireless sensor and actuator networks (WSANs). Indeed, many WSANs applications, including node localization, sleep schedule, and data aggregation, require accurate time synchronization to function properly. In this thesis, we propose two cluster-based time synchronization protocols for WSANs, namely Sensor and Actuator Networks Synchronization Protocol (SANSync), and Optimized Sensor and Actuator Networks Synchronization Protocol (OSANSync). These protocols, contrary to existing protocols, fully exploit the available resources of the actuators, particularly their large transmission range, to improve time synchronization accuracy. We also propose a heuristic-based method to select the ROOT node through which all the other nodes in the network are synchronized. The proposed method is fully distributed and can be easily integrated into time synchronization protocols to improve their performance.

Keywords: Time synchronization, Wireless sensor and actuator networks, Clustering, Heuristic-based methods.

Résumé

Le but de la synchronisation temporelle est de permettre aux horloges des différents noeuds d'un réseau d'indiquer des valeurs relativement proches à tout moment. Actuellement, la synchronisation temporelle est un problème fondamental dans les réseaux de capteurs et actionneurs sans fil (WSAN). En effet, de nombreuses applications des WSANs, notamment la localisation des noeuds, la planification de mise en veille et l'agrégation des données, nécessitent une synchronisation temporelle précise pour fonctionner correctement. Dans cette thèse, nous proposons deux protocoles de synchronisation temporelle pour les WSANs basés sur le clusterisation du réseau, à savoir "Sensor and Actuator Networks Synchronization Protocol (SANSync)", et "Optimized Sensor and Actuator Networks Synchronization Protocol (OSANSync)". Ces protocoles, contrairement aux protocoles existants, exploitent pleinement les ressources disponibles des actionneurs, en particulier leur grande portée de transmission, pour améliorer la précision de la synchronisation temporelle. Nous proposons également une méthode heuristique pour sélectionner le noeud racine à partir duquel tous les autres noeuds du réseau sont synchronisés. La méthode proposée est entièrement distribuée, et peut être facilement intégrée dans les protocoles de synchronisation temporelle pour améliorer leurs performances.

Mots-clefs: Synchronisation temporelle, Réseaux de capteurs avec actionneurs, Clusterisation, Méthodes heuristiques.

ملخص الأطروحة

الغرض من مزامنة الوقت هو تمكين مختلف ساعات الأجهزة الموجودة في شبكة معينة من عرض قيم قريبة نسبيًا من بعضها البعض في أي لحظة. في الوقت الحالي ، تعد مزامنة الوقت مشكلة أساسية في شبكات الاستشعار اللاسلكية (WSANs). في الواقع ، كثير من تطبيقات WSANs ، بما في ذلك تحديد مواقع الأجهزة و برمجة توقف الأجهزة و تجميع البيانات ، تتطلب تزامنًا دقيقًا للوقت لتعمل بشكل صحيح. في هذه الأطروحة ، نطور طريقتين لمزامنة الوقت في شبكات WSANs ، وهما بروتوكول التزامن SANSync ، و البروتوكول المحسنOSANSync. تعمل البروتوكولات المقترحة على تجميع الأجهزة في حزم ، وخلافًا للبروتوكولات الحالية ، فإنها تستغل بشكل كامل الموارد المتاحة للأجهزة الموجودة في الشبكة ، ولا سيما نطاق الإرسال الكبير ، لتحسين مزامنة الوقت بشكل كبير. نقترح أيضًا طريقة لتحديد الجهاز المركزي التي تتم من خلاله مزامنة جميع الأجهزة الأخرى في الشبكة. الطريقة المقترحة هي موزعة بالكامل ويمكن دمجها بسهولة في بروتوكولات مزامنة الوقت لتحسين أدائهم.

الكلمات المفتاحية: مزامنة الوقت ، شبكات الاستشعار اللاسلكية، تجميع الأجهزة.